Eléments de robotique avec Arduino

Pascal MASSON

(pascal.masson@unice.fr)

Version projection
Edition 2019-2020-V12

Les Moteurs

POLYTECH NICE-SOPHIA

École Polytechnique Universitaire de Nice Sophia-Antipolis
Parcours des écoles d'ingénieurs Polytech
1645 route des Lucioles, 06410 BIOT

Université Nice SOPHIA ANTIPOLIS

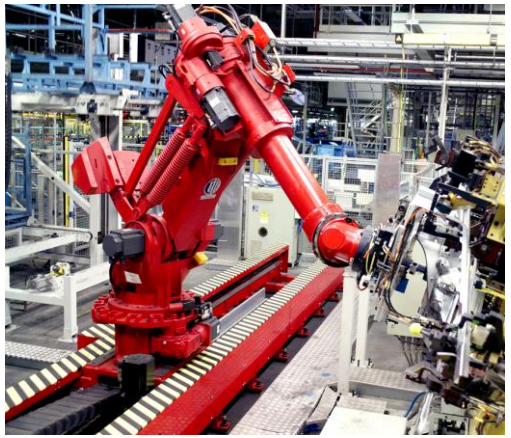
Sommaire

EN COURS D'ECRITURE

Introduction

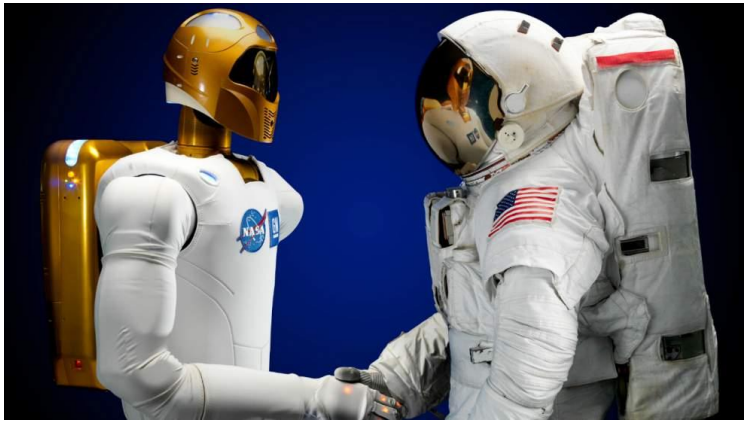
Pourquoi des moteurs ?

- Les moteurs électriques permettent de convertir l'énergie électrique en mouvement mécanique
- Ils sont un des éléments majeurs de la robotique puisqu'ils permettent les mouvements et/ou déplacements. C'est pour cela qu'on les retrouve dans tous les robots



Robot industriel

Robot démineur



Robonaute 2

Introduction

Types de moteurs

- On distingue plusieurs types de moteurs électriques mais ne traitera dans ce cours que de 4 moteurs :

  - ✓ Moteur à courant continu avec ou sans réducteur (qui permet de modifier le ratio couple – vitesse)

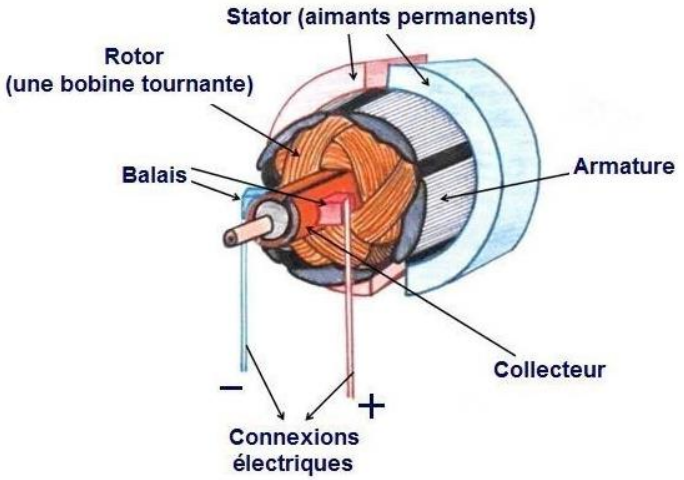
  - ✓ Servomoteur

  - ✓ Moteur pas à pas

  - ✓ Moteur Brushless

1.1. Description

- Il est constitué de deux parties électriques : le stator et le rotor. Lorsqu'on alimente le moteur, il se crée une interaction magnétique qui met le moteur en mouvement. Lorsqu'on inverse le sens de la tension qui alimente le moteur, il tourne en sens inverse.



- Dit autrement, un moteur CC transforme une tension continue en un couple
- Un système de frottement permet d'alimenter le rotor : des balais (ou charbons montés, « brush » en Anglais) frottent sur les contacts en rotation : le collecteur.
- On utilise en général des moteurs à excitation constante. L'inducteur est formé d'aimants permanents. Seul le rotor est alimenté par 2 fils. Lorsqu'on inverse la polarité du moteur, il tourne dans le sens inverse.

1. Moteur CC

1.1. Description

- Les moteurs se caractérisent essentiellement par leur couple (en N.m ou oz.in ou kg.cm), leur vitesse de rotation (en Rotation Par Minute, rpm) et leur tension d'alimentation

7000 rpm
12 V
0.54 N.m

19300 rpm
12 V
0.53 N.m

66000 rpm
3 V

1. Moteur CC

1.1. Description

- Pour gagner en couple, il est possible de connecter un réducteur (engrenages) de vitesse sur l'arbre du moteur

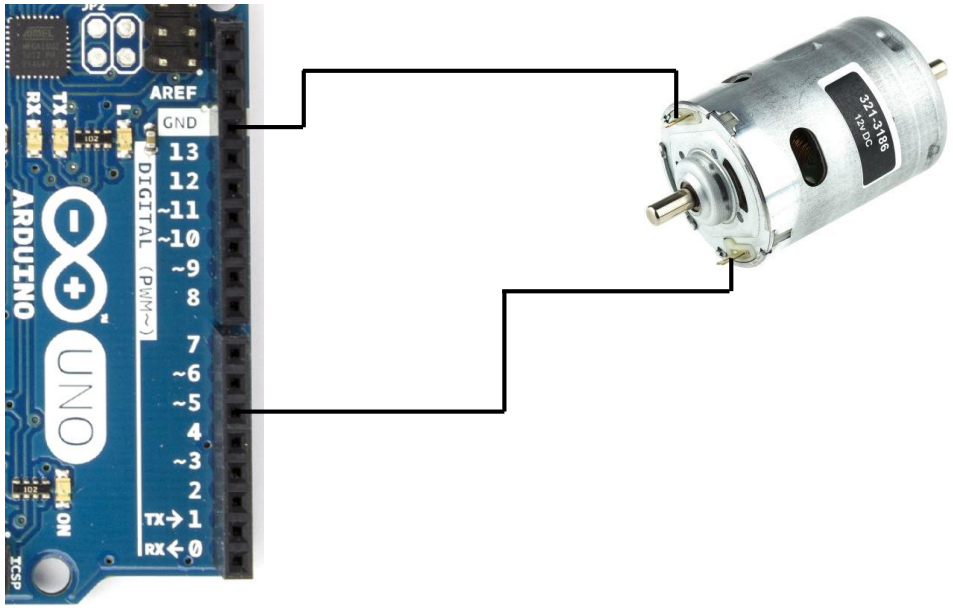
1750 rpm
12 V
0.21 N.m

67 rpm
12 V
2 N.m

27 rpm
12 V
2.45 N.m

1.2. Alimentation direct par arduino : A NE JAMAIS FAIRE

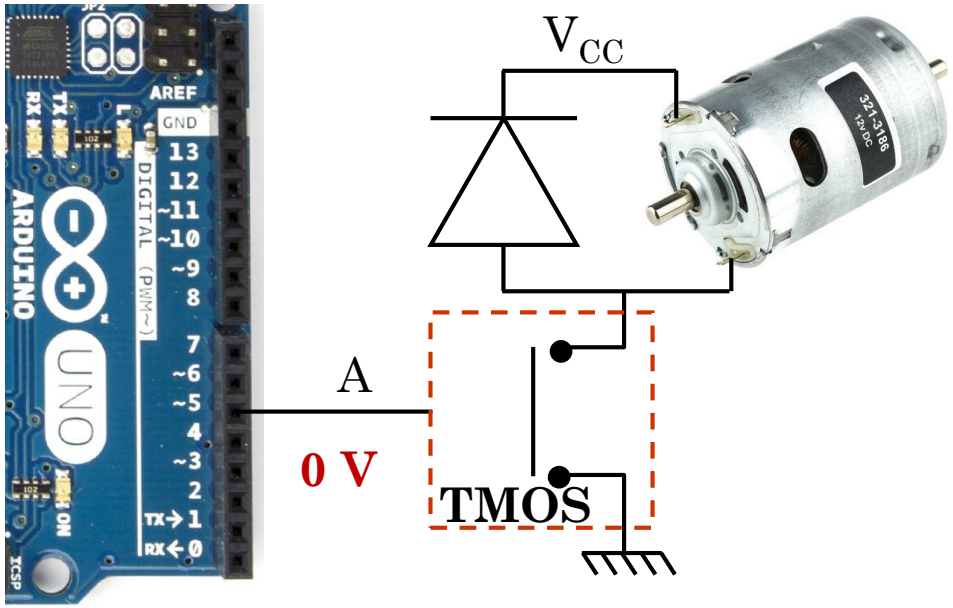
- Si on décide d'alimenter un moteur CC directement avec l'arduino, il risque de se passer 2 choses :
  - ✓ L'arduino n'arrivera pas à fournir suffisamment de courant
  - ✓ Quand la sortie sera coupée, l'énergie emmagasinée dans la bobine du moteur risque de détruire la carte



1.3. Alimentation via un MOSFET

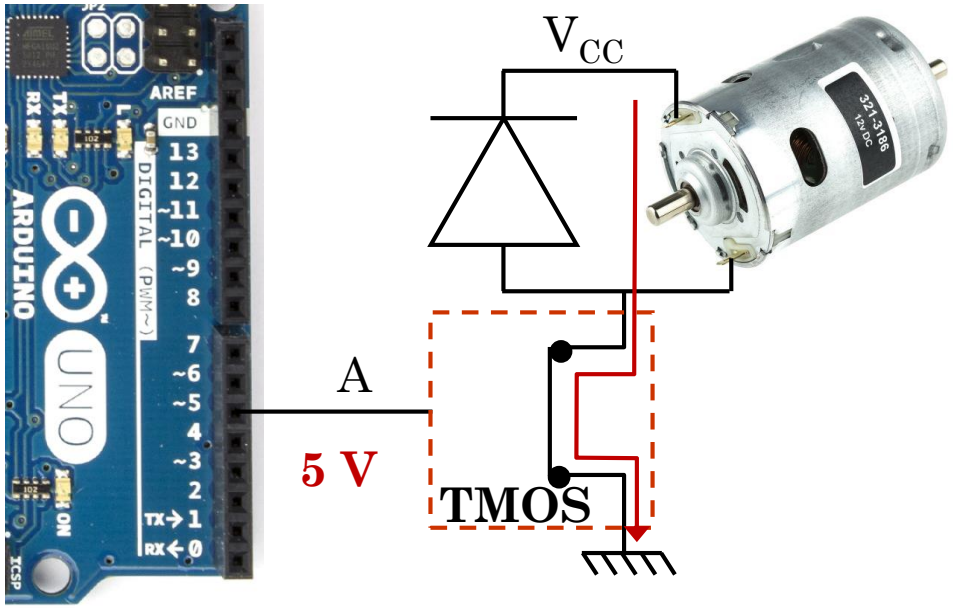
Fonctionnement du montage

- Nous allons utiliser un transistor MOSFET qui servira de commande pour le moteur
- V_{CC} correspond à la tension d'alimentation du moteur qui peut être différente du 5 V de l'arduino
- La masse de l'arduino et celle du générateur VCC sont connectées ensembles
- Quand la sortie (ici n° 5) de l'arduino fournie 0 V, le transistor MOSFET se comporte comme un interrupteur ouvert donc le moteur ne tourne pas.



1.3. Alimentation via un MOSFET

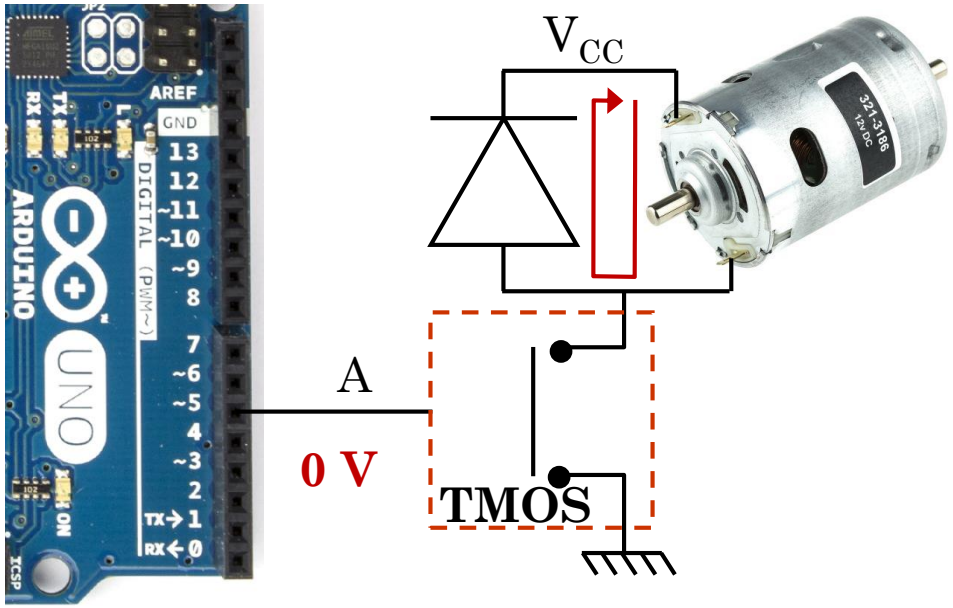
- **Fonctionnement du montage**
  - Quand la sortie de l'arduino fournie 5 V, le transistor MOSFET se comporte comme un interrupteur fermé donc le moteur tourne.
  - La bobine du moteur emmagasine de l'énergie



1.3. Alimentation via un MOSFET

Fonctionnement du montage

- Quand la sortie de l'arduino fournie 5 V, le transistor MOSFET se comporte comme un interrupteur fermé donc le moteur tourne.
- La bobine du moteur emmagasine de l'énergie
- Si on applique à nouveau 0 V sur le transistor, l'interrupteur s'ouvre et l'énergie emmagasinée s'évacue par la diode
- Cette diode est appelée « diode de roue libre » et elle évite d'avoir des surtensions sur le transistor et de le griller et par suite de griller la carte



1.3. Alimentation via un MOSFET

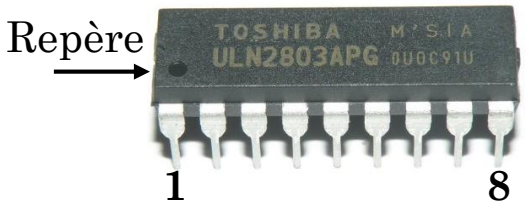
Réglage de la vitesse du moteur

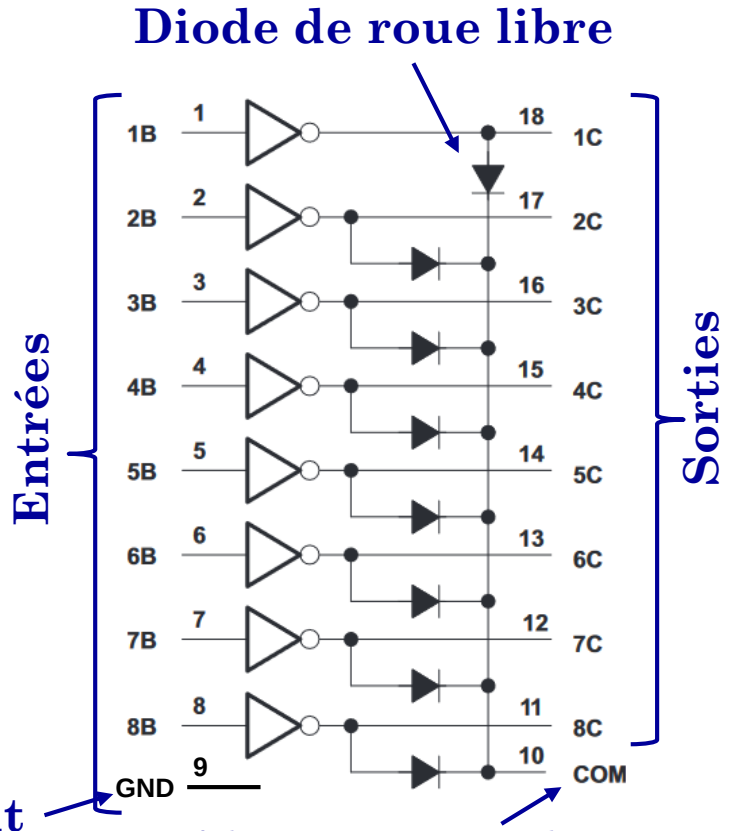
- Comme il n'est pas possible de fournir une tension analogique afin de régler la résistance de l'interrupteur (entre circuit ouvert et court-circuit), on peut utiliser un signal PWM
- A chaque fois que A = 5 V, la tension V_{CC} fournie de l'énergie au moteur pour qu'il tourne. La fréquence du PWM est suffisamment grande pour la rotation du moteur ne soit pas saccadée
- Si le signal PWM est toujours nul, le moteur ne tourne pas
- Si le signal PWM est toujours égal à 5 V, le moteur tourne a sa vitesse maximale

1.4. Alimentation via un ULN2803

Présentation du circuit

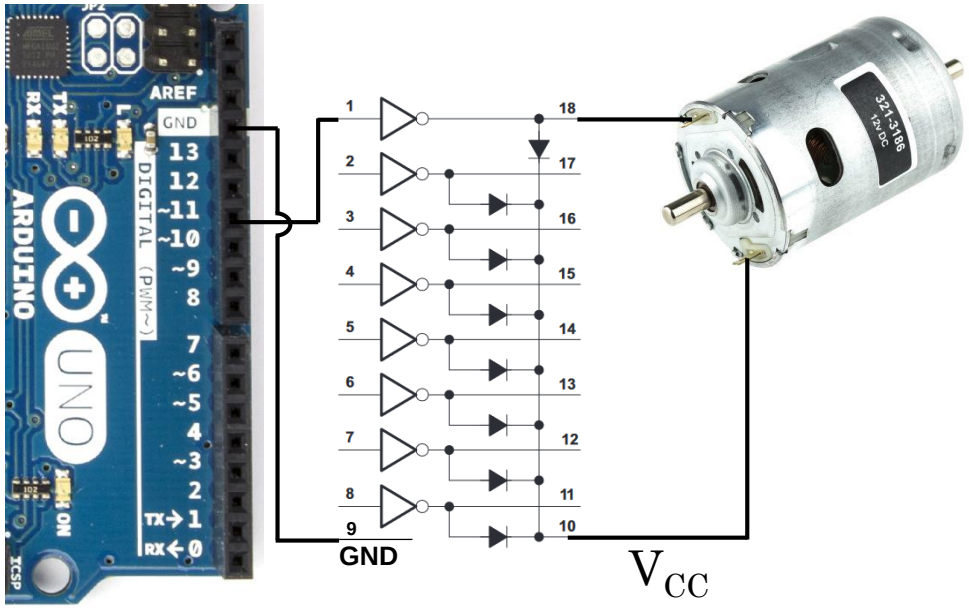
- L'ULN2803 se compose de 8 transistors Darlington NPN (combinaison de deux **transistors** bipolaires).
- Le courant collecteur est au maximum de 500 mA mais les Darlingtons peuvent mis en parallèle pour augmenter ce courant
- Une diode de roue libre est associée à chaque sortie





1.4. Alimentation via un ULN2803

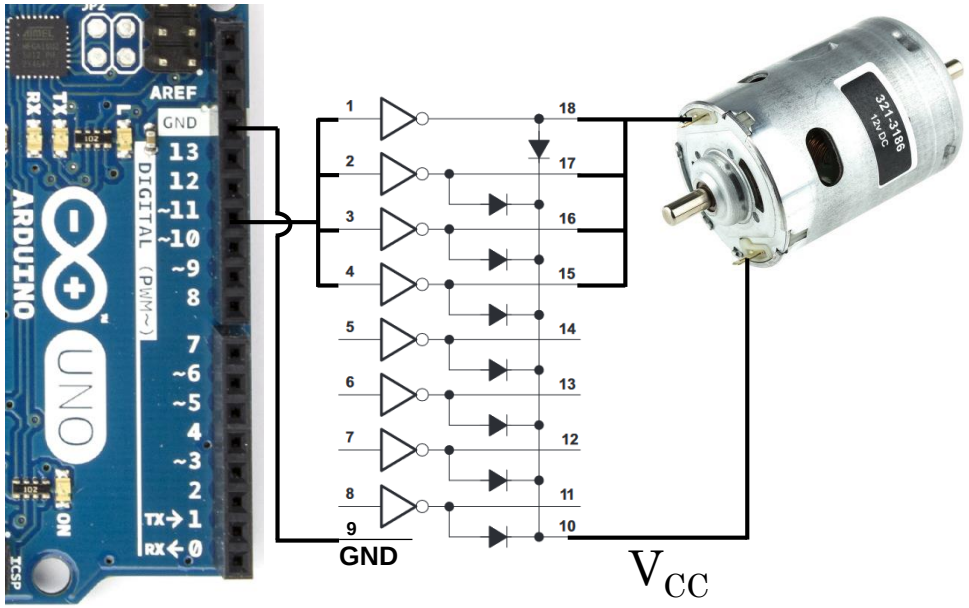
- **Exemple d'utilisation**
  - Il ne faut pas oublier de relier ensemble les masses de l'arduino et du générateur de tension V_{CC} qui alimente le moteur
  - Il faut vérifier que la consommation du moteur est compatible avec les 500 mA que peut fournir le Darlington sous peine de griller le circuit



1.4. Alimentation via un ULN2803

Exemple d'utilisation

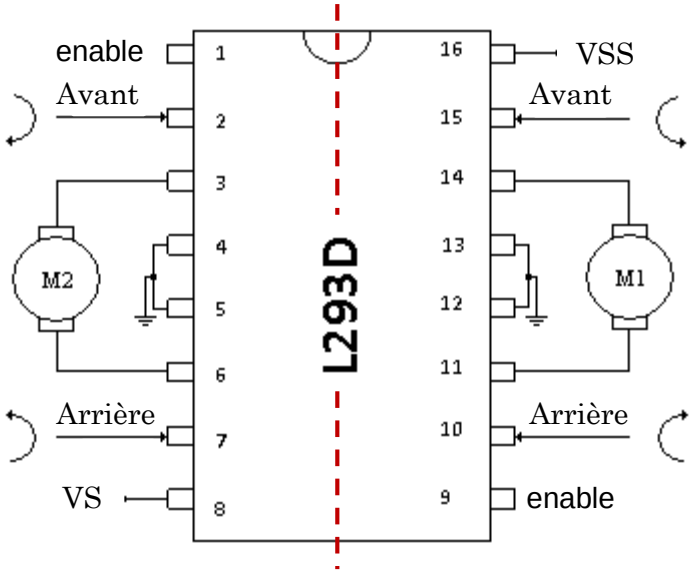
- Il ne faut pas oublier de relier ensemble les masses de l'arduino et du générateur de tension V_{CC} qui alimente le moteur
- Il faut vérifier que la consommation du moteur est compatible avec les 500 mA que peut fournir le Darlington sous peine de griller le circuit
- On peut aussi utiliser plusieurs sortie pour augmenter le courant
- Le moteur ne tourne que dans un sens ce qui convient, par exemple, dans le cas d'une pompe à air ou à eau

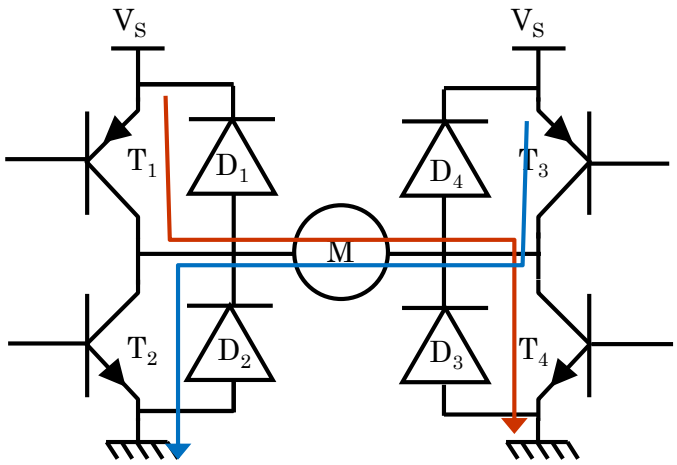


1.5. L293 – quadruple demi pont en H

Présentation du circuit

- Ce circuit permet de réaliser 2 ponts en H donc d'alimenter 2 moteurs qui peuvent tourner dans les 2 sens
- La tension d'alimentation des moteurs est comprise entre 4.3 et 36 V
- Le courant d'alimentation du moteur est de 1 A (600 mA pour le L293D) mais le circuit supporte des piques de 2 A (1.2 A pour le L293D)

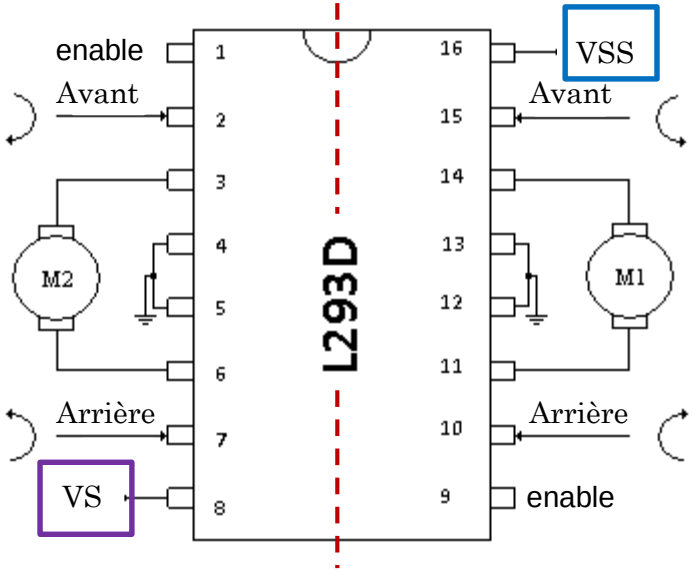


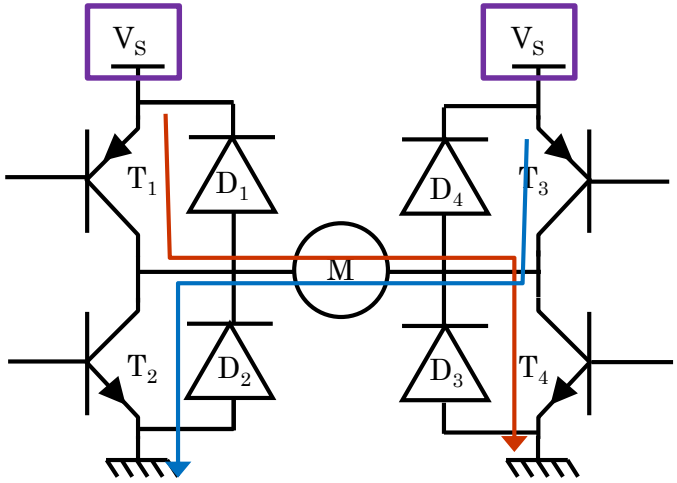


1.5. L293 – quadruple demi pont en H

Présentation du circuit

- Des entrées «enable» permettent d'activer la commande des moteurs
- Il y a une alimentation V_{SS} pour la partie logique et les transistors et une alimentation V_S pour le moteur.
- Le pont en H se compose de 2 transistors NPN et de 2 transistors PNP sui fonctionnent en saturé/bloqué

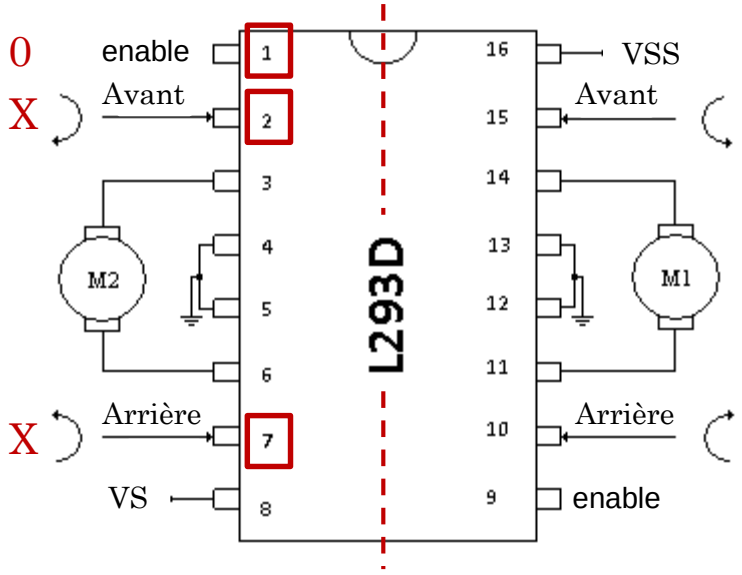


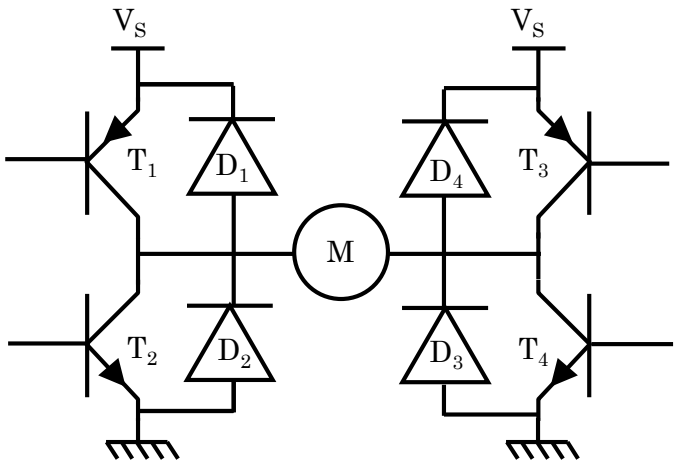


1.5. L293 – quadruple demi pont en H

Fonctionnement du montage

- Si on applique un 0 logique sur l'entrée « enable » (n°1), les 4 transistors sont bloqués et donc équivalents à des interrupteurs ouverts quelles que soient les valeurs (X) données aux entrées n°2 et n°7

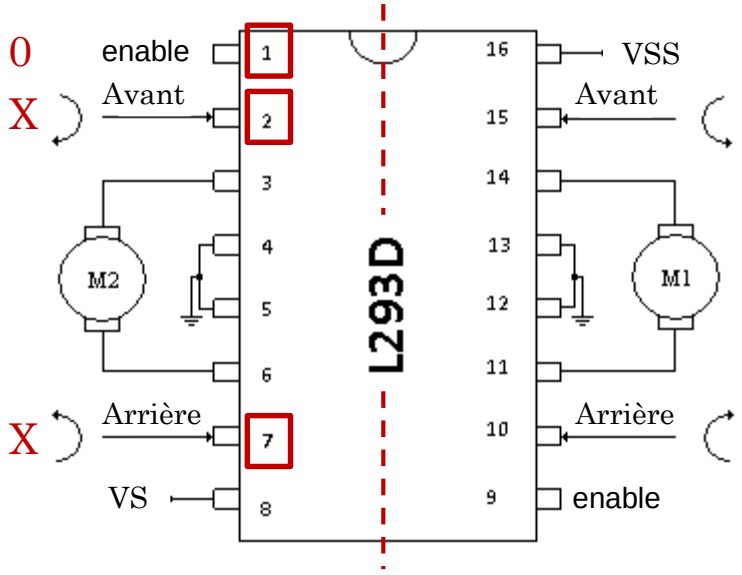


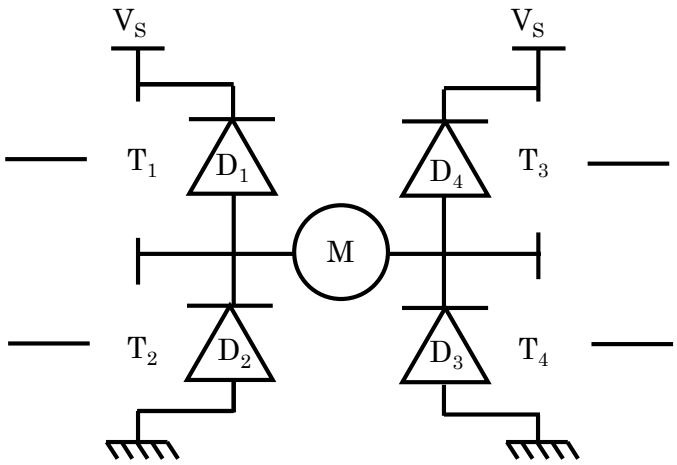


1.5. L293 – quadruple demi pont en H

Fonctionnement du montage

- Si on applique un 0 logique sur l'entrée « enable » (n°1), les 4 transistors sont bloqués et donc équivalents à des interrupteurs ouverts quelles que soient les valeurs données aux entrées n°2 et n°7
- On en conclut qu'il n'y pas de courant qui circule dans le moteur et qu'il ne tourne pas

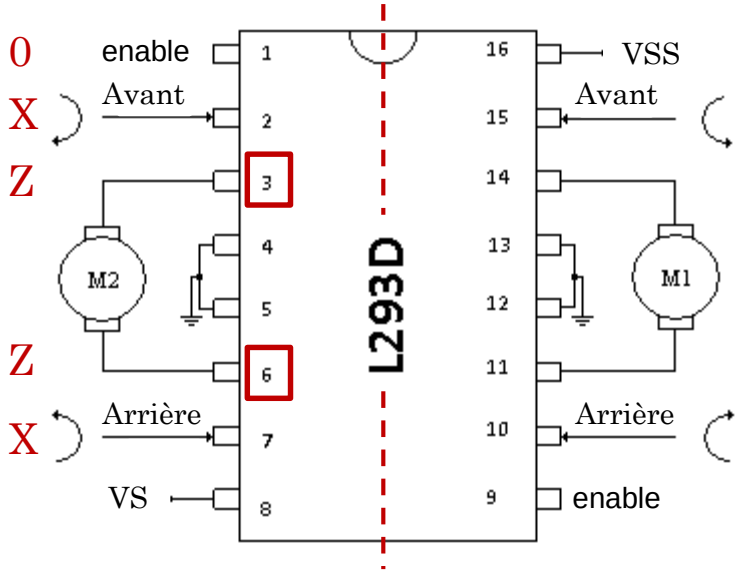


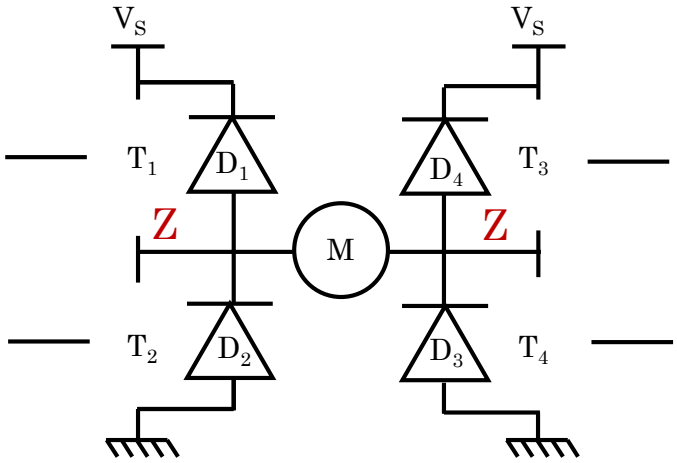


1.5. L293 – quadruple demi pont en H

Fonctionnement du montage

- Les sorties n°3 et n°6 sont en haute impédance que l'on note Z

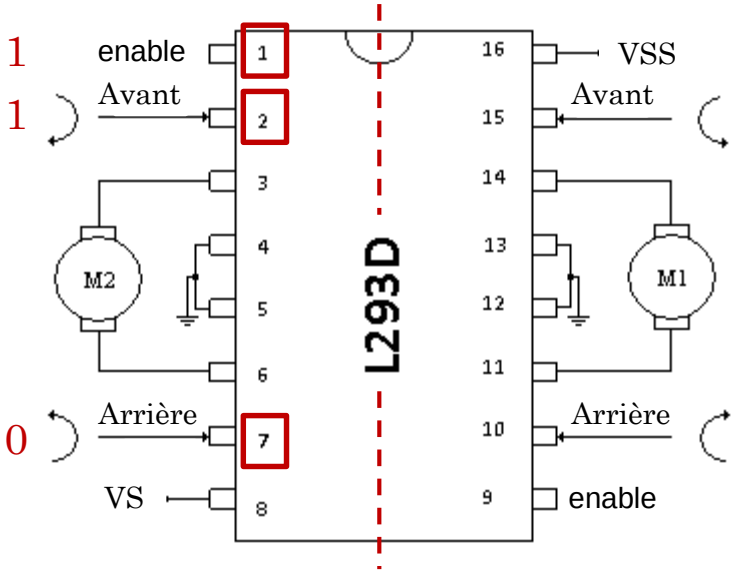


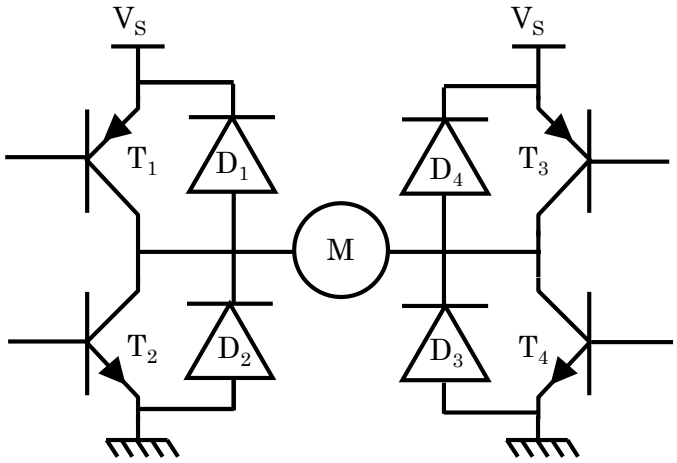


1.5. L293 – quadruple demi pont en H

Fonctionnement du montage

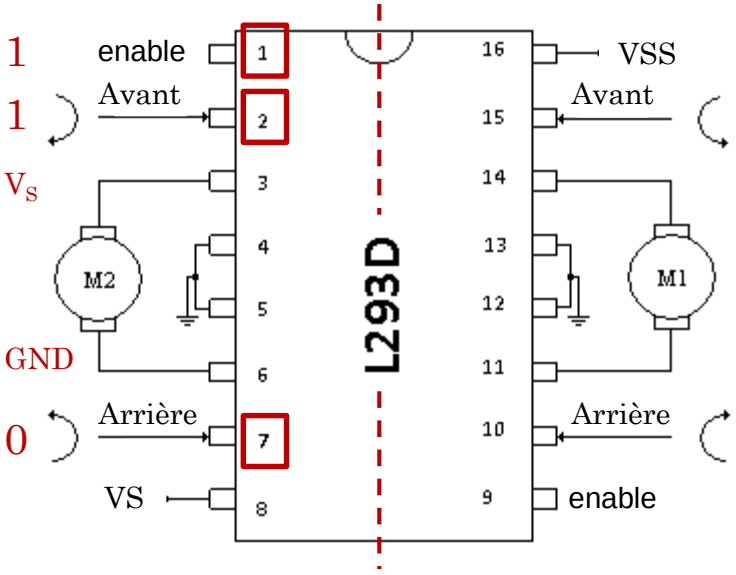
- On applique un 1 logique sur l'entrée « enable » et sur l'entrée n°2 et un 0 logique sur l'entrée n°7
- Les transistors T_2 et T_3 sont bloqués alors que T_1 et T_4 sont saturés et se comportent comme des courts-circuits

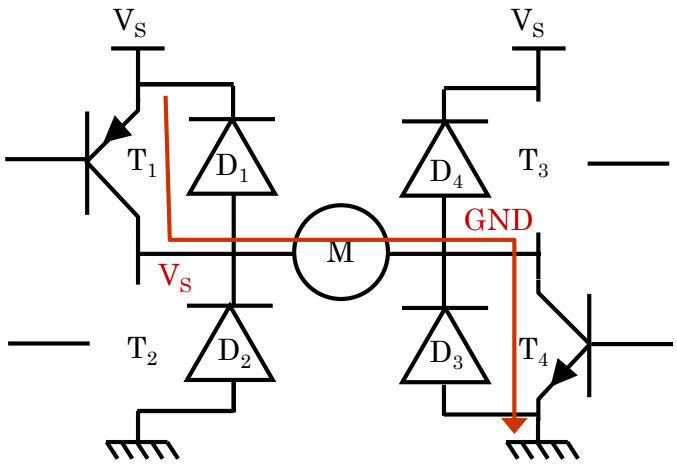




1.5. L293 – quadruple demi pont en H

- **Fonctionnement du montage**
  - T_1 et T_4 amène V_S et la masse aux bornes du moteur qui se met à tourner

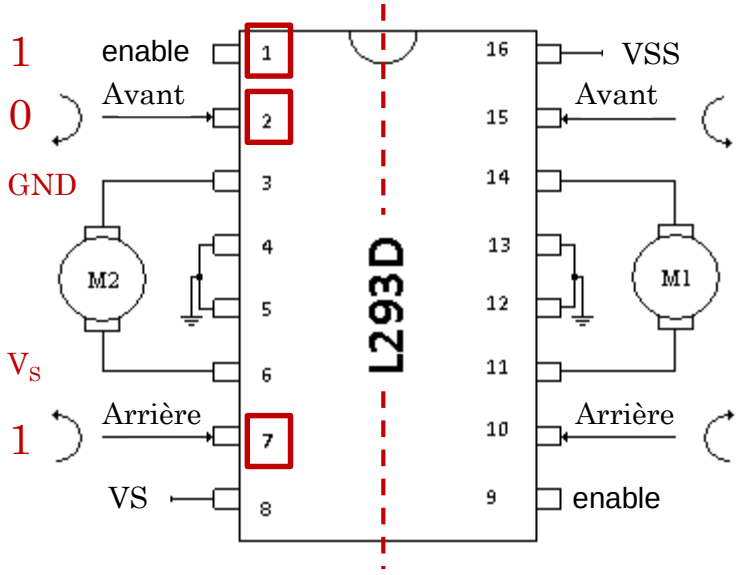


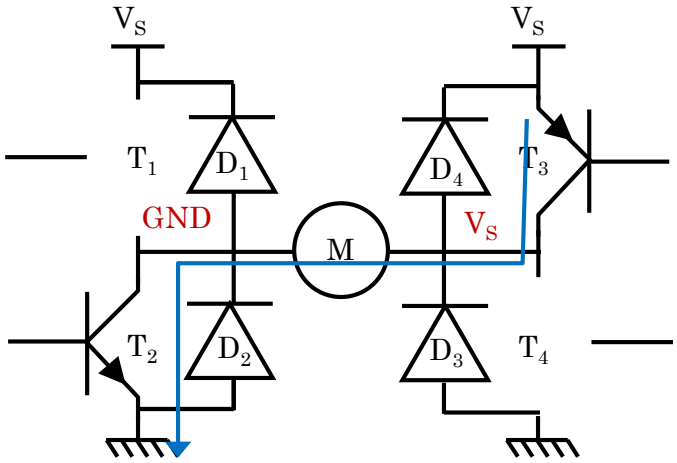


1.5. L293 – quadruple demi pont en H

Fonctionnement du montage

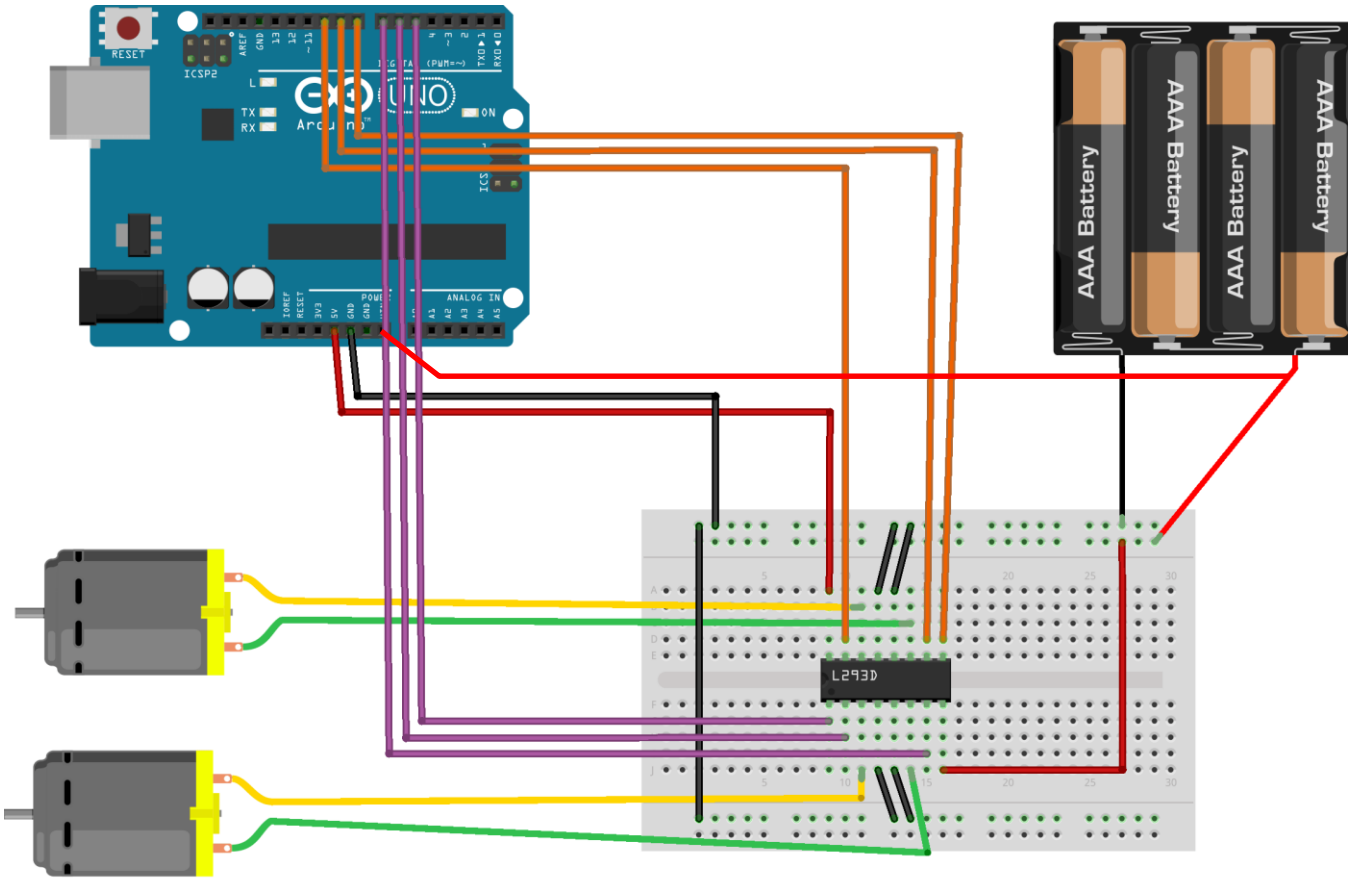
- Si on met l'entrée n°2 à 0 et l'entrée n°7 à 1, alors le courant change de sens dans le moteur
- L'entrée enable peut être utilisée en PWM ce qui permet d'avoir une variation allant de 0 (rotation nulle) à 255 (rotation maximale) pour le moteur dans les deux sens de rotation





1.5. L293 – quadruple demi pont en H

- **Exemple de montage avec 2 moteurs**

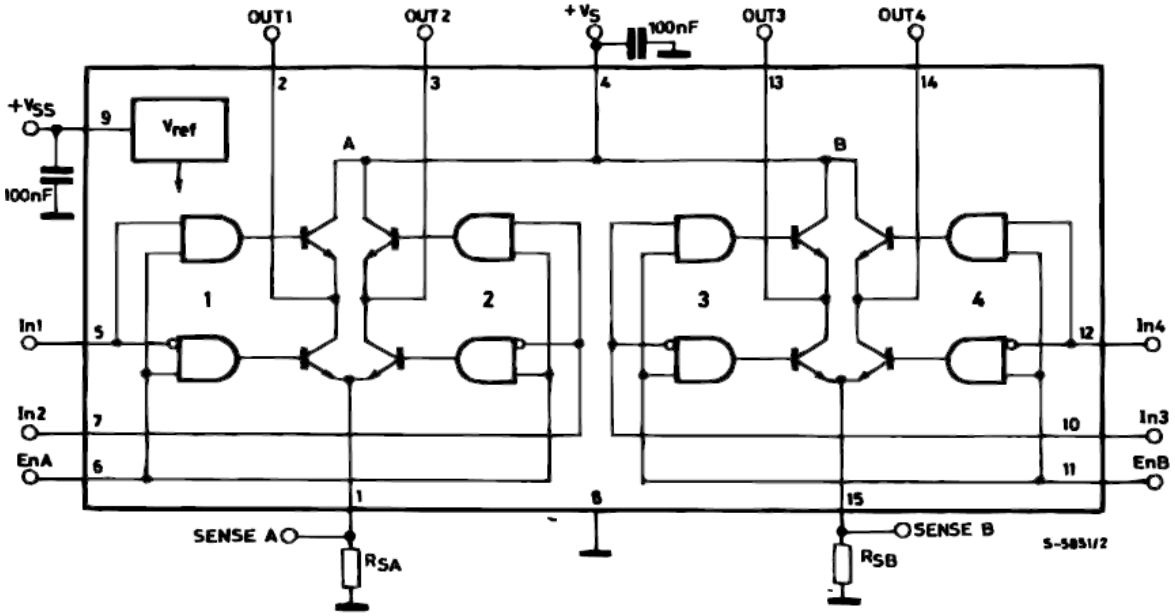


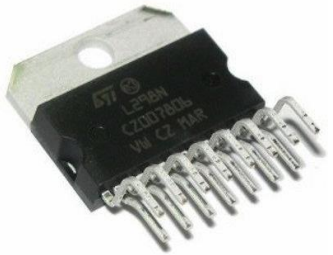
blog.whatgeek.com.pt/2015/03/arduino-l293d-dc-motors-control/

1.6. L298 – quadruple demi pont en H

Présentation du circuit

- C'est le même composant que le L293 mais il permet un courant de 2 A (avec des piques de 3 A) et il peut être vissé sur un radiateur pour le refroidir
- La tension d'alimentation du moteur peut aller jusqu'à 50 V
- Attention : les diodes de roue libre ne sont pas intégrées au circuit

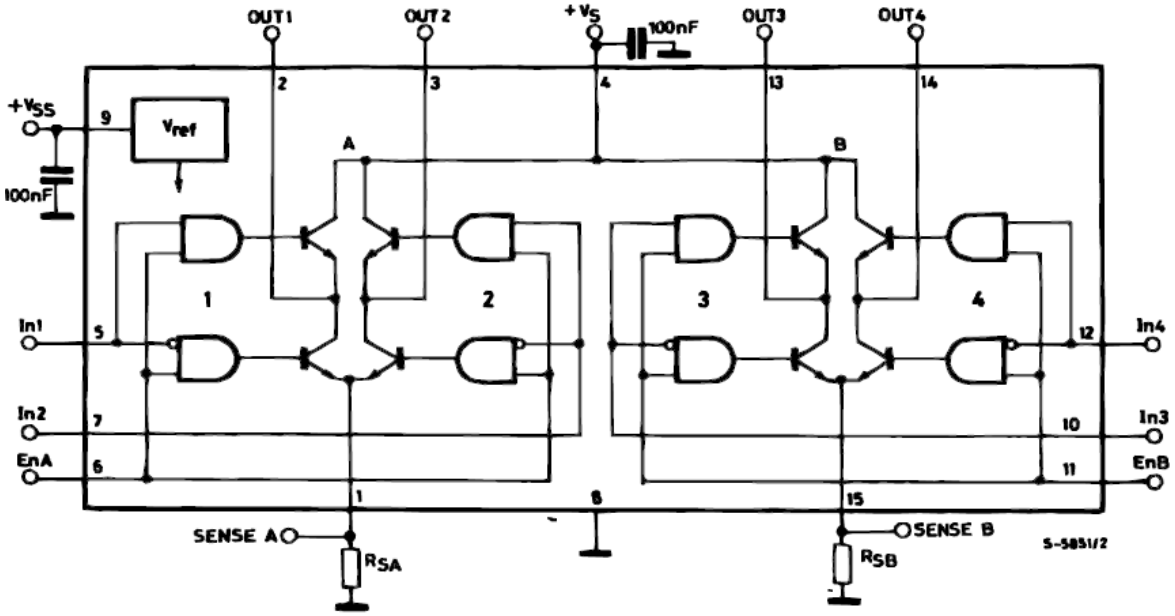




1.6. L298 – quadruple demi pont en H

Présentation du circuit

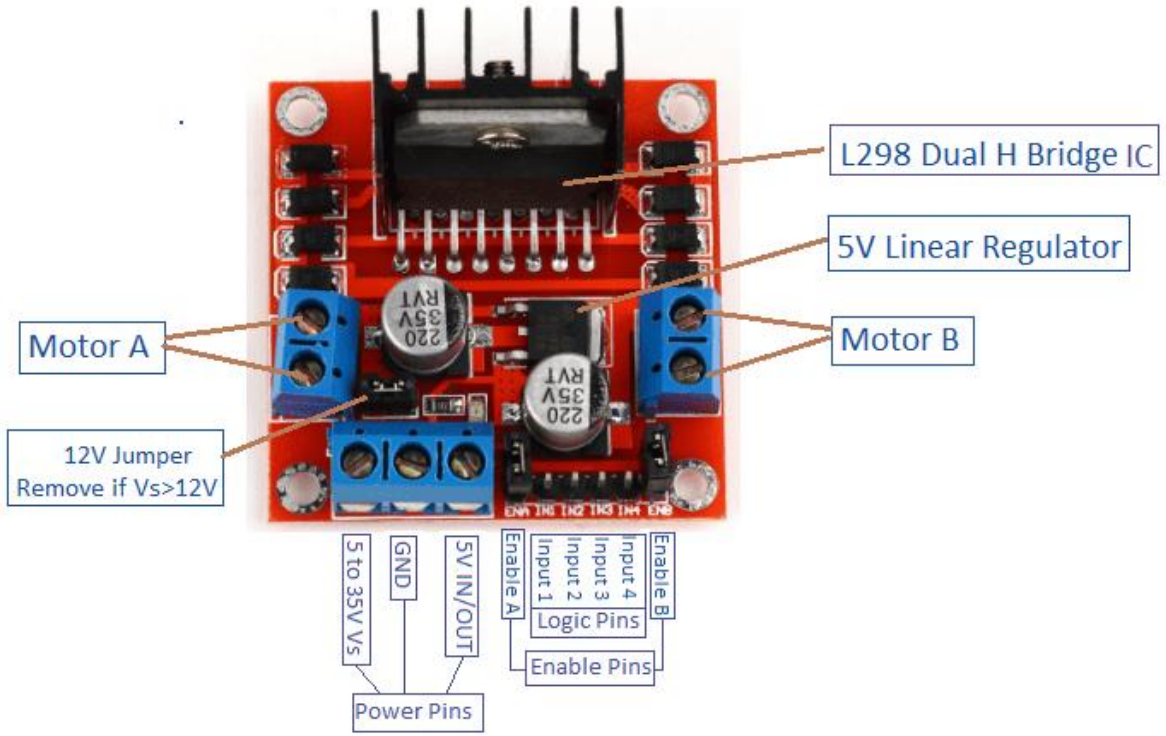
- L'entrée enable peut être utilisée en PWM ce qui permet d'avoir une variation allant de 0 (rotation nulle) à 255 (rotation maximale) pour le moteur dans les deux sens de rotation



1.6. L298 – quadruple demi pont en H

Utilisation du circuit

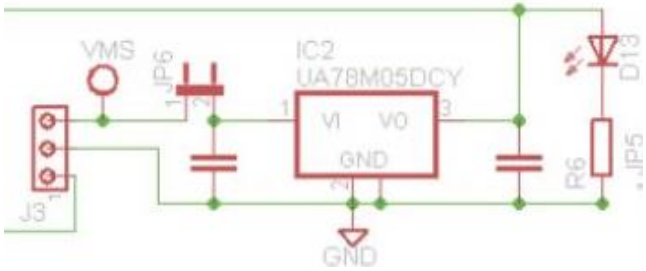
- On peut utiliser des modules qui intègrent déjà tous les composants qui accompagnent le circuit comme les diodes de roue libre
- La plage d'alimentation pour les moteurs va de 5 à 35 V



1.6. L298 – quadruple demi pont en H

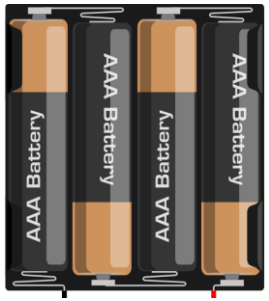
Utilisation du circuit

- Le module délivre aussi une tension de 5 V qui permet d'alimenter le L298 et la carte arduino.



- Pour cela, le module utilise le régulateur « uA78M05 » dont la tension d'entrée doit être dans la gamme [7V, 25V].

- Si on utilise un bloc de 4 piles de 1.5V (donc 6 V au total) avec la voiture du cours par exemple, la tension de 5 V en sortie ne sera pas garantie d'une part et d'autre part, elle risque de s'écrouler quand on tire du courant sur les piles pour faire tourner les moteurs (surtout quand les piles sont usées).



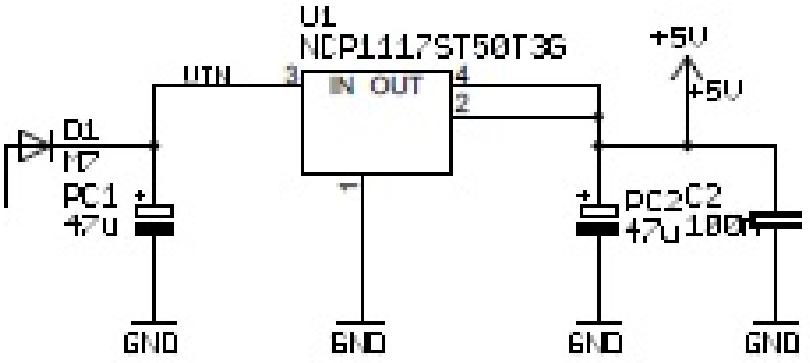
1.6. L298 – quadruple demi pont en H

- **Utilisation du circuit**
  - Dans ce cas, le L298 peut présenter un comportement anormal (un des 2 ponts qui ne marche pas ou des vitesses réduites par exemples) quand les signaux appliqués aux entrées de commandes (état « 1 ») par la carte arduino sont inférieurs à la tension d'alimentation de la puce.
  - Même si les piles sont parfaitement chargées, ce phénomène est observable quand on laisse la carte arduino branchée sur une autre source de 5 V (le PC par exemple) : les moteurs ne démarrent pas !

1.6. L298 – quadruple demi pont en H

Utilisation du circuit

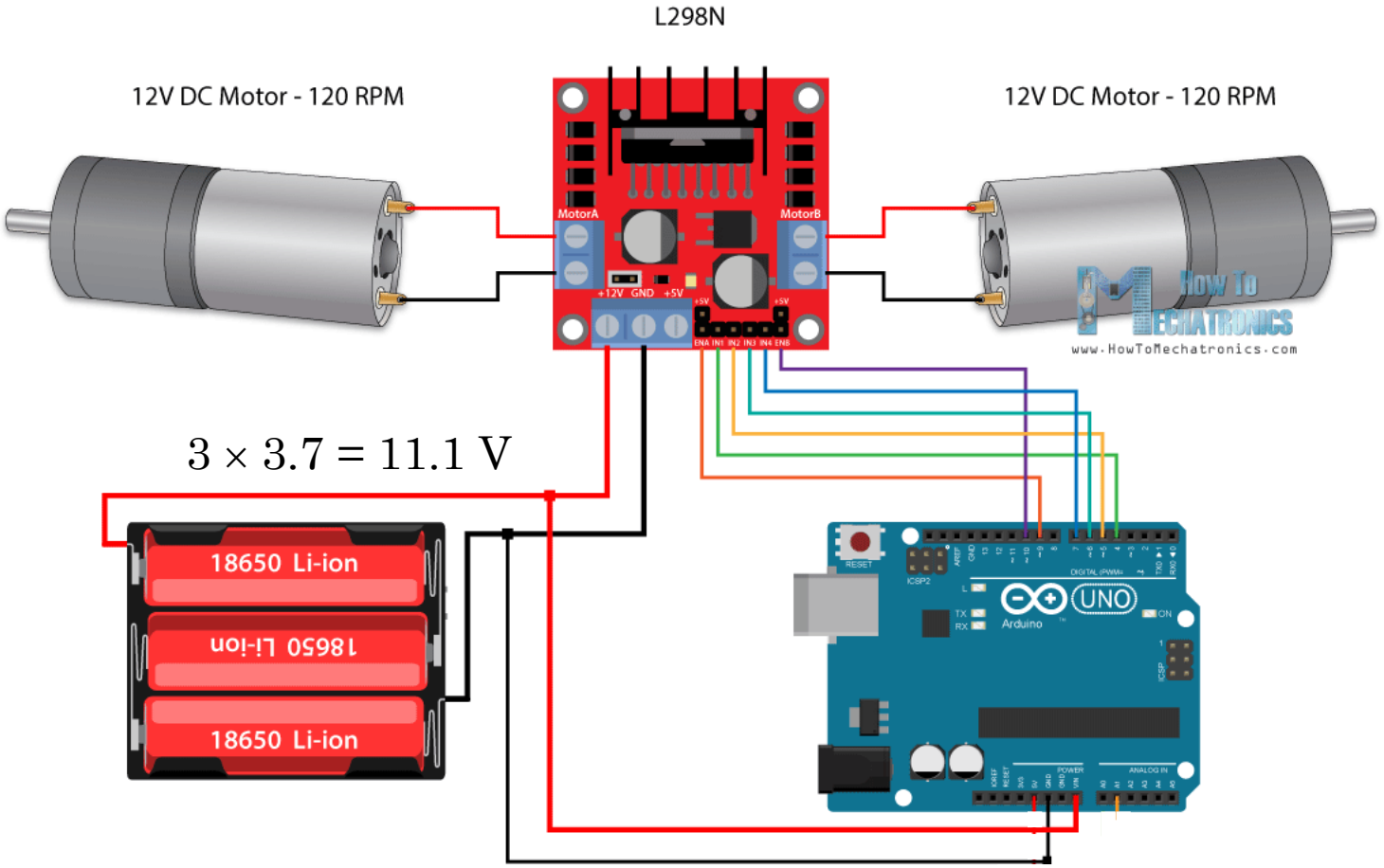
- Le régulateur de tension utilisé par la carte arduino est le NCP1117ST503G qui accepte la gamme de tensions « normale » [7V, 12V]. La gamme limite de tensions est [6.5 (voir 6V), 20 V].



- Donc le régulateur de la carte arduino est plus performant que celui du L298. En conséquence, si on branche les piles directement sur le Vin de la carte arduino, le L298 peut présenter des aberrations de comportement.
- En conclusion et afin d'éviter tout problème, il est préférable d'alimenter la carte arduino avec la batterie et d'alimenter le module avec le 5 V de la carte arduino. Il faut évidement penser à enlever le jumper JP6 du module.
- Dans l'idéal, il faut une batterie pour alimenter les moteurs et une autre pour générer le 5 V (les masses doivent être communes.

1.6. L298 – quadruple demi pont en H

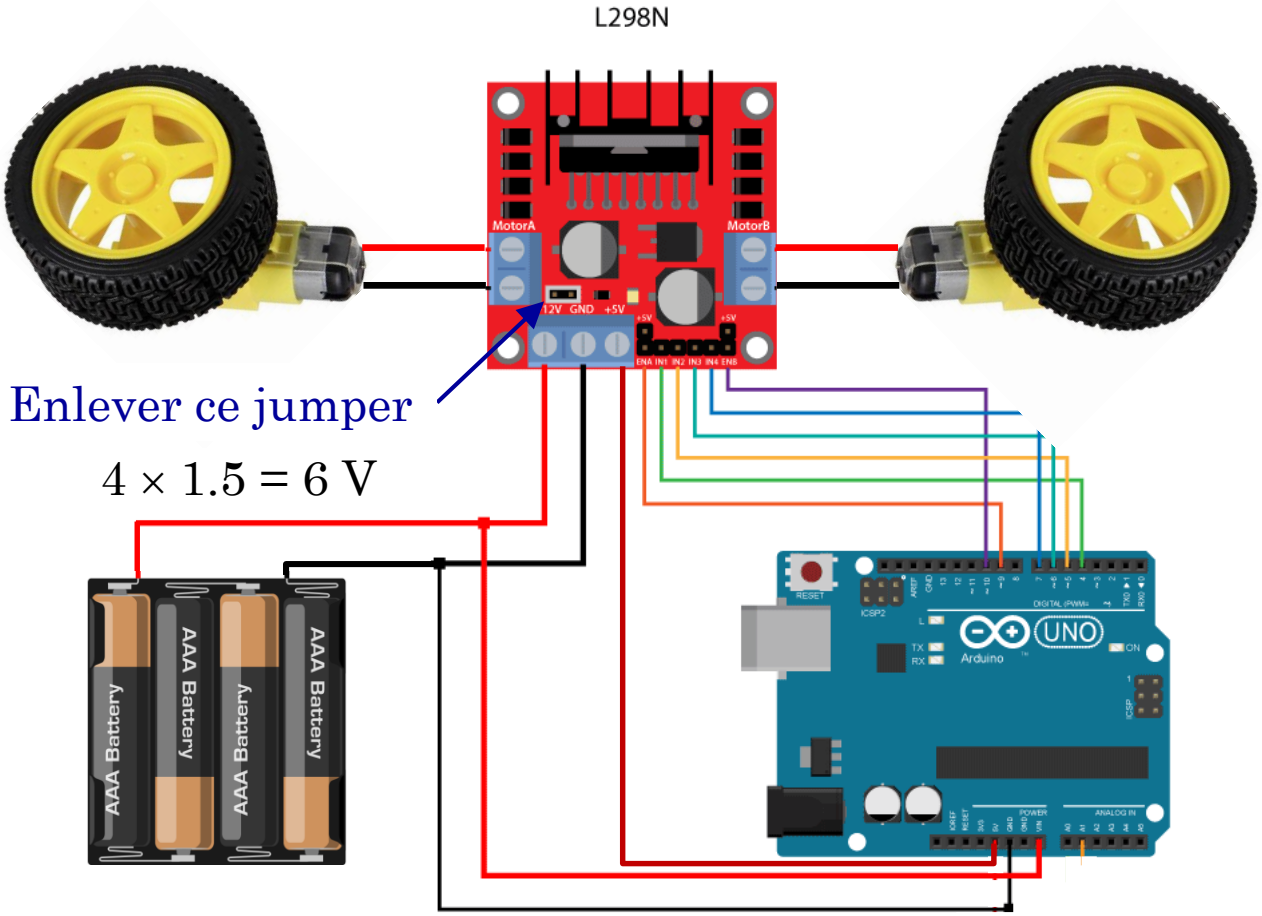
- **Exemple de montage avec 2 moteurs (alimentation > 7 V)**



howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/

1.6. L298 – quadruple demi pont en H

- **Exemple de montage avec 2 moteurs (alimentation ≈ 6 V)**



howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/

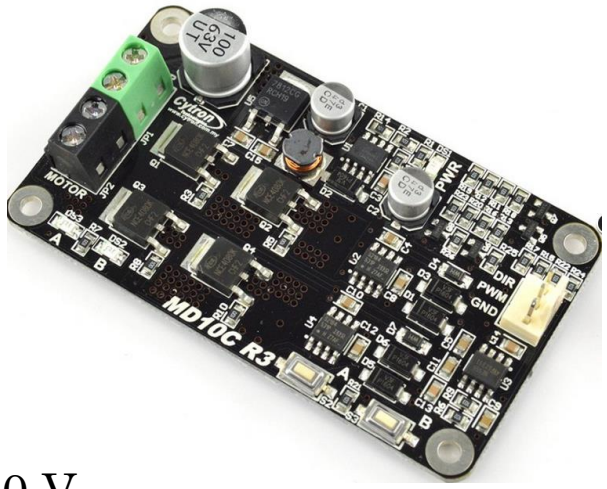
1.6. L298 – quadruple demi pont en H

Exemple de code pour 2 moteurs

```
//-- MOTEUR A --
int ENA=9; //Connecté à Arduino pin 9(sortie PWM)
int IN1=4; //Connecté à Arduino pin 4
int IN2=5; //Connecté à Arduino pin 5

//-- MOTEUR B --
int ENB=10; //Connecté à Arduino pin 10(Sortie PWM)
int IN3=6; //Connecté à Arduino pin 6
int IN4=7; //Connecté à Arduino pin 7

void setup() {
 pinMode(ENA,OUTPUT);   // Configurer
 pinMode(ENB,OUTPUT);   // les broches
 pinMode(IN1,OUTPUT);    //  comme sortie
 pinMode(IN2,OUTPUT);
 pinMode(IN3,OUTPUT);
 pinMode(IN4,OUTPUT);
 digitalWrite(ENA,LOW);// Moteur A - Ne pas tourner
 digitalWrite(ENB,LOW);// Moteur B - Ne pas tourner
// Direction du Moteur A
 digitalWrite(IN1,LOW);
 digitalWrite(IN2,HIGH);

 // Direction du Moteur B
 // NB: en sens inverse du moteur
A
 digitalWrite(IN3,HIGH);
 digitalWrite(IN4,LOW);
}

void loop() {
  // Moteur A - Plein régime
  analogWrite(ENA,255);

  // Moteur B - Mi-régime
  analogWrite(ENB,128);
}
```

1.7. Contrôleur de 1 moteur « Cytron »

Présentation du circuit



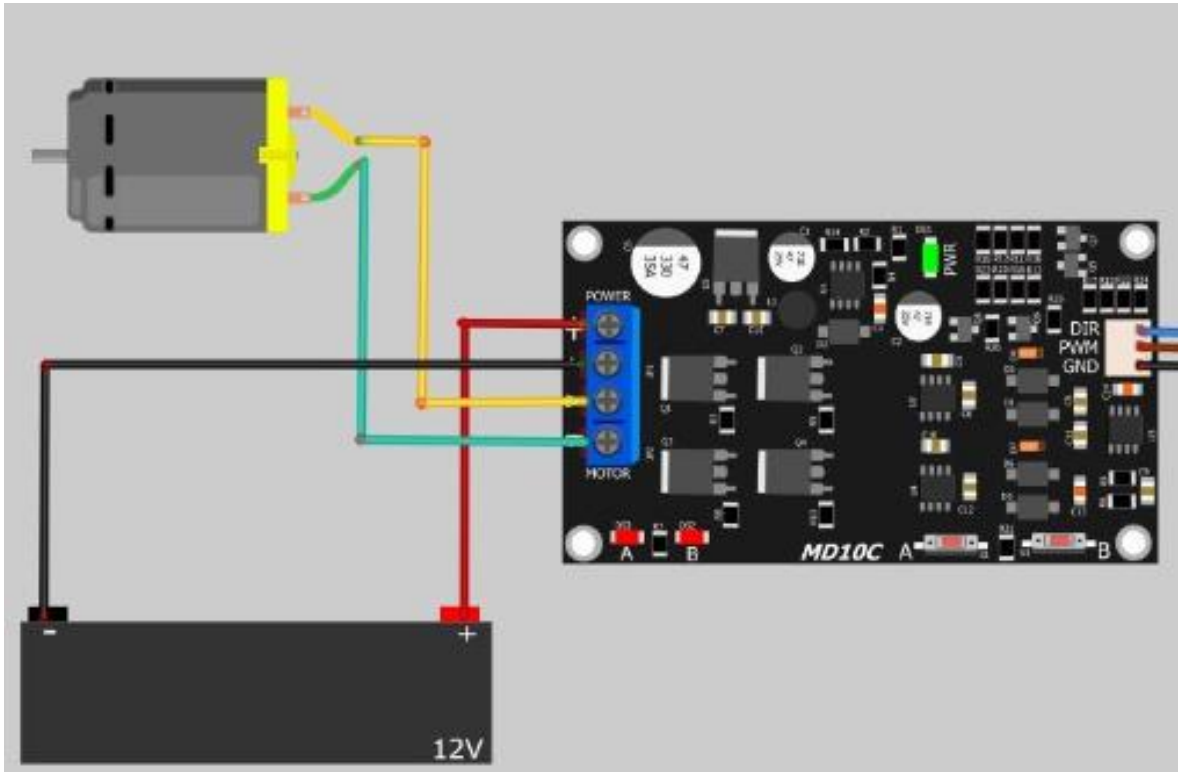
- Il existe des contrôleurs qui permettent d'alimenter un moteur avec beaucoup plus de courant comme avec le contrôleur Cytron :

  - ✓ Courant continu de 13 A
  - ✓ Tension d'alimentation du moteur de 5 à 30 V
  - ✓ Le pont en H utilise des transistors MOS qui consomment beaucoup moins que les bipolaires ce qui explique l'absence de radiateur

1.7. Contrôleur de 1 moteur « Cytron »

Utilisation

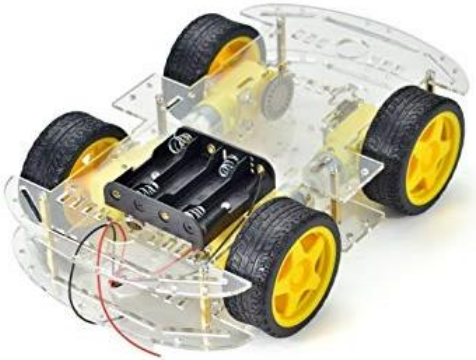
- Contrairement aux circuits L29X, ce contrôleur n'a une seule entrée pour indiquer le sens de rotation



GND
PWM pour la vitesse
Sens de rotation

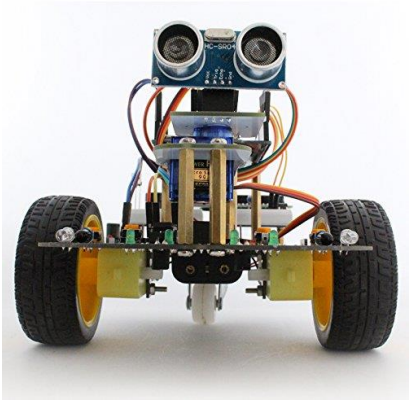
1.8. Choix d'un moteur

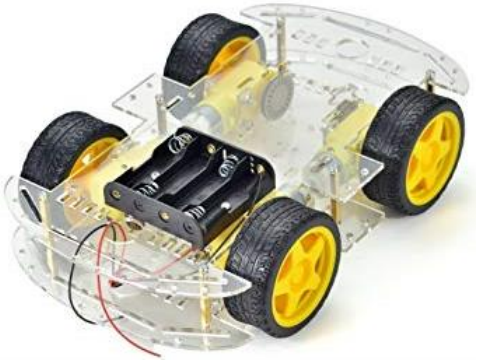
- **Paramètres à prendre en compte**
  - Les paramètres du robot sont :
    - ✓ Masse du robot



1.8. Choix d'un moteur

- **Paramètres à prendre en compte**
  - Les paramètres du robot sont :
    - ✓ Masse du robot
    - ✓ Nombre de roues





1.8. Choix d'un moteur

- **Paramètres à prendre en compte**
  - Les paramètres du robot sont :
    - ✓ Masse du robot
    - ✓ Nombre de roues
    - ✓ Rayon des roues

1.8. Choix d'un moteur

- **Paramètres à prendre en compte**
  - Les paramètres du robot sont :
    - ✓ Masse du robot
    - ✓ Nombre de roues
    - ✓ Rayon des roues
    - ✓ Tension de la batterie

3.7 V

9 V

11.7 V

12 V

24 V

1.8. Choix d'un moteur

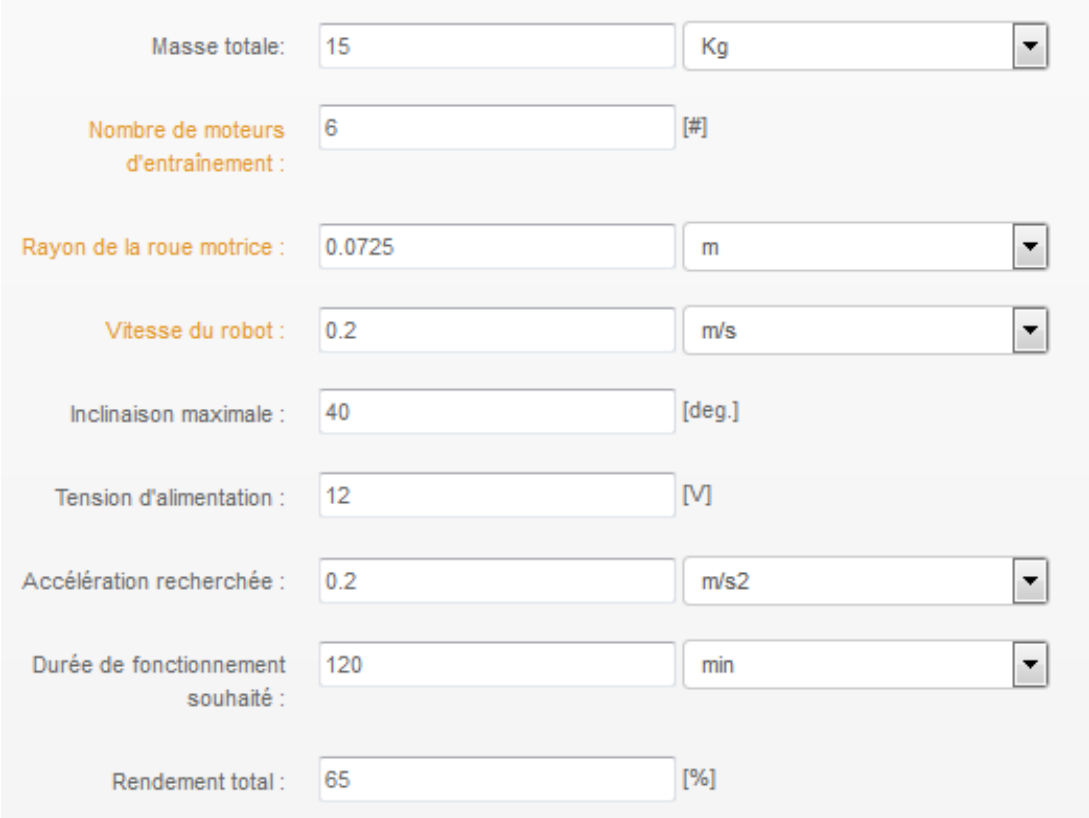
- **Paramètres à prendre en compte**
  - Les paramètres du robot sont :
    - ✓ Masse du robot
    - ✓ Nombre de roues
    - ✓ Rayon des roues
    - ✓ Tension de la batterie
  - Il faut aussi prendre en compte :
    - ✓ Le temps de fonctionnement
    - ✓ La vitesse et l'accélération
    - ✓ L'inclinaison maximale du parcours

1.8. Choix d'un moteur

Outil de calcul

- Une multitude de sites internet propose des outils de dimensionnement des moteurs et nous nous basons sur celui du site « robotshop.com »

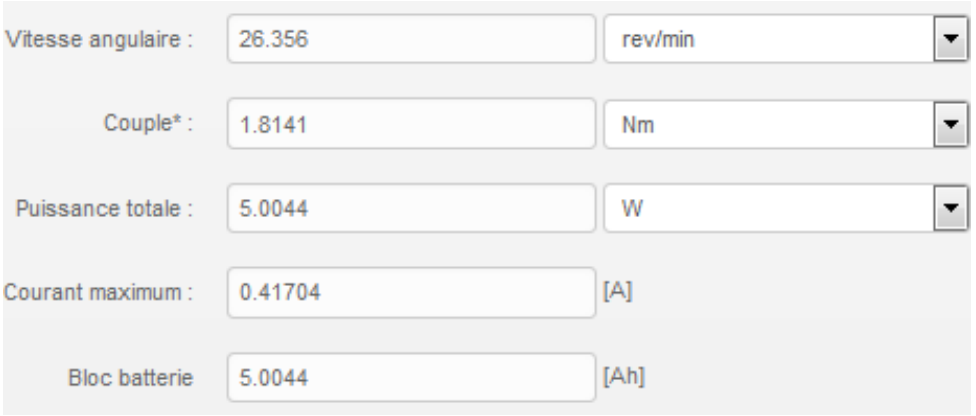
- Les paramètres sont ceux du rover martien de Polytech Nice, R6P4 (users.polytech.unice.fr/~pmasson/rover.php)



1.8. Choix d'un moteur

Outil de calcul

- Le site donne alors
  - ✓ La vitesse de rotation des moteurs
  - ✓ Le couple des moteurs
  - ✓ La charge de la batterie

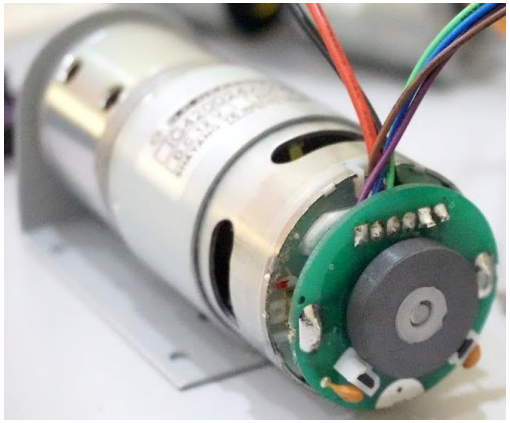


- Il reste alors à trouver le moteur dont les caractéristiques s'approchent de celles calculées

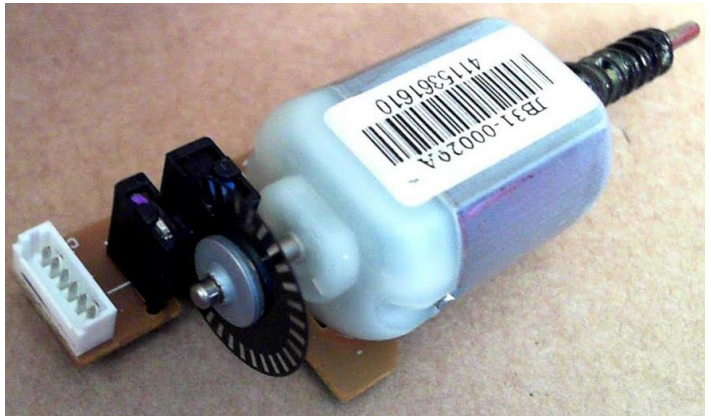
1.9. Mesure de la vitesse de rotation

Présentation des encodeurs

- En robotique mobile, les encodeurs rotatifs sont utilisés pour mesurer le déplacement (sens et vitesse de rotation) de chacune des roues du robot
- Un encodeur permet d'obtenir une information en quasi-temps réel sur la position et vitesse du moteur
- Les encodeurs sont soit optiques (émission réception de lumière) soit magnétiques avec un capteur à effet hall



Effet Hall

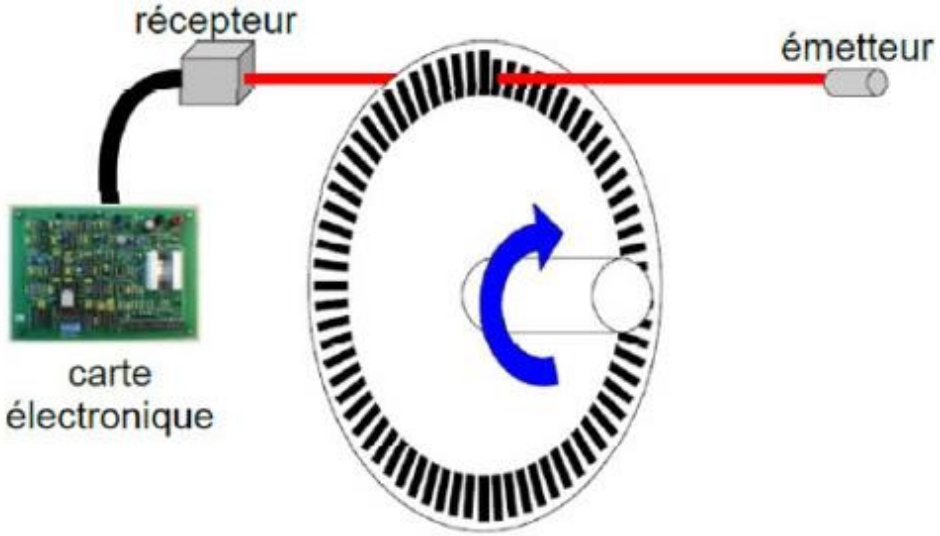


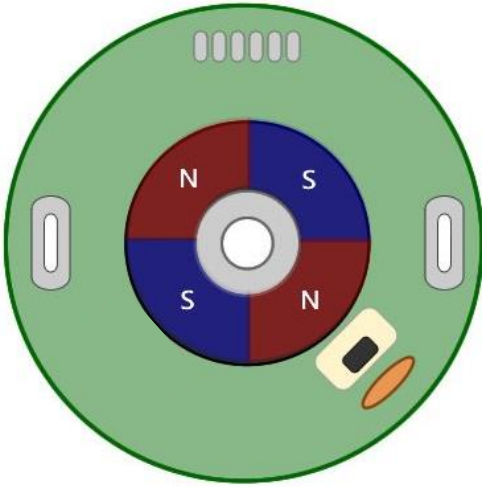
Optique

1.9. Mesure de la vitesse de rotation

Présentation des encodeurs

- La lumière émise par une diode IR est reçue ou non par un phototransistor en fonction de la position de la roue codeuse





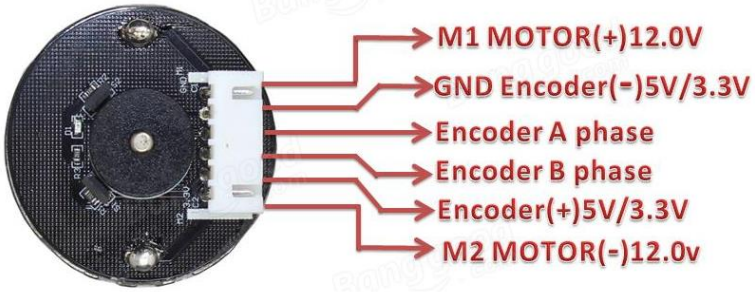
- Des capteurs à effet hall détectent le passage des pôles nord et sud d'aimants

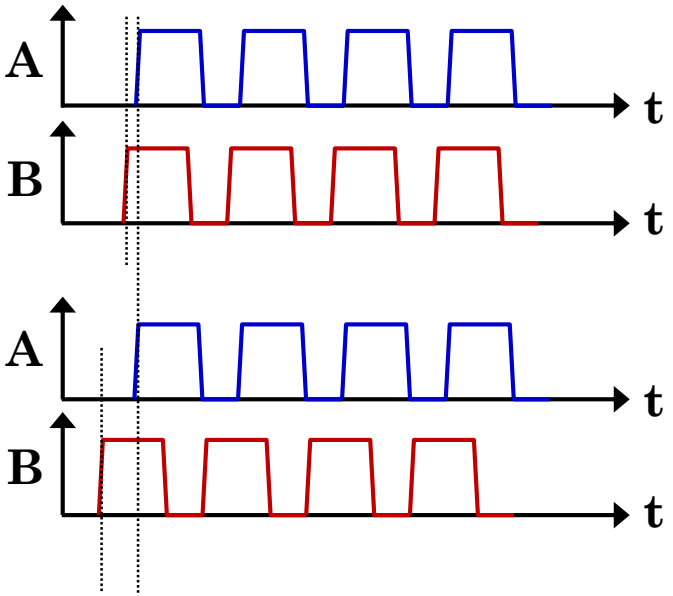
1.9. Mesure de la vitesse de rotation

Présentation des encodeurs

- Un encodeur à effet hall peut aussi être constitué de 2 capteurs à effet hall ce qui permet de déterminer le sens de rotation du moteur

- Le décalage temporel entre l'apparition des phases A et B dépend du sens de rotation

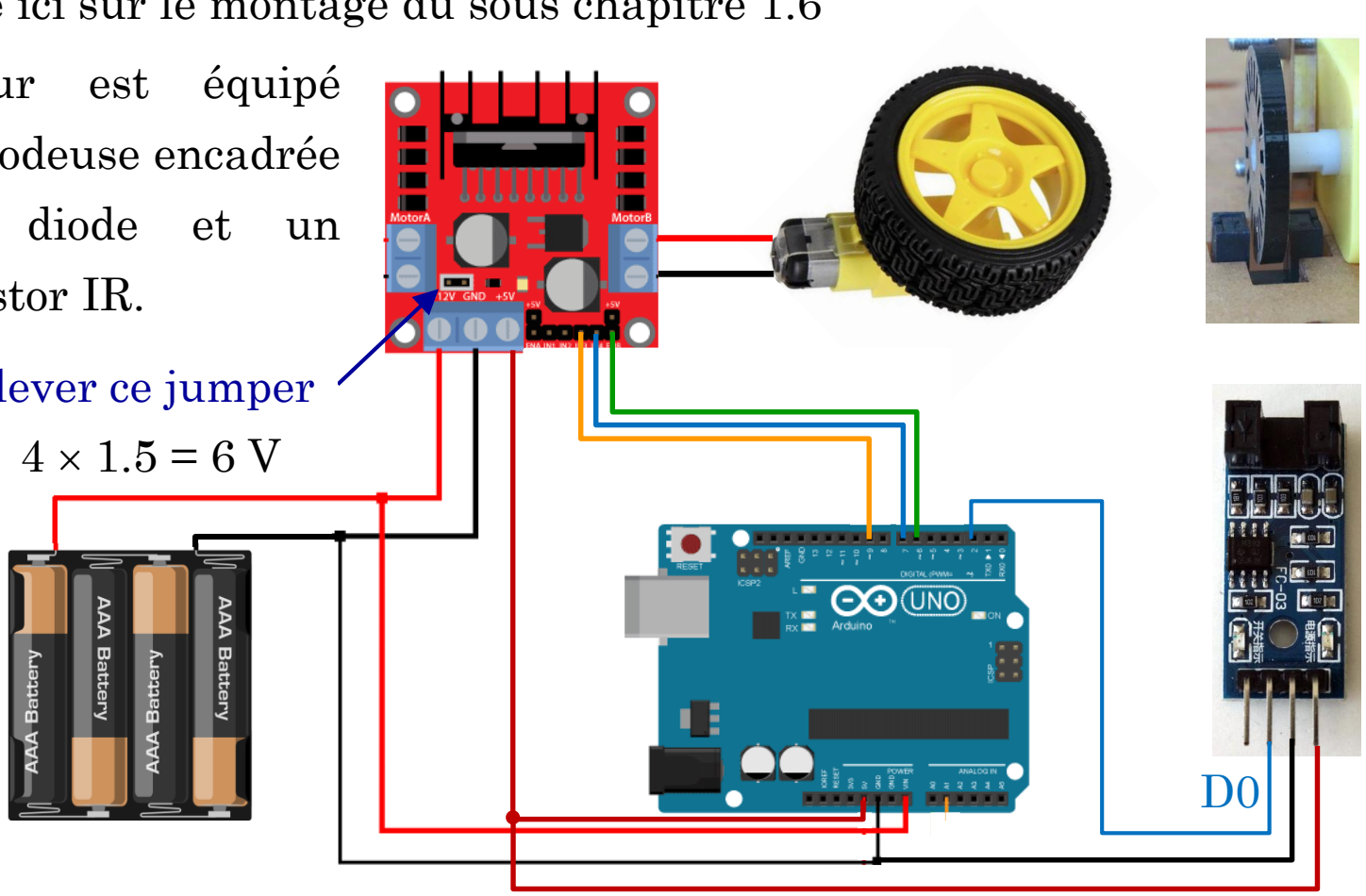




1.9. Mesure de la vitesse de rotation

Utilisation du capteur IR

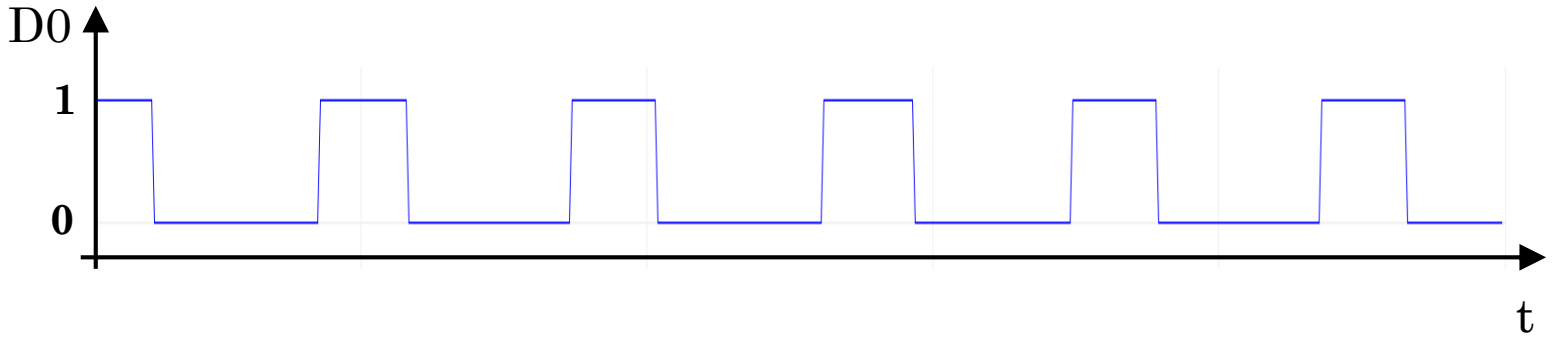
- On se base ici sur le montage du sous chapitre 1.6
- Le moteur est équipé d'une roue codeuse encadrée pour une diode et un phototransistor IR.



1.9. Mesure de la vitesse de rotation

Utilisation du capteur IR

- La roue codeuse présente 20 fentes
- Voici la variation de l'IO n°2 (D0) en fonction du temps (traceur série) lorsque l'on fait tourner le moteur :



- Il n'y a pas de symétrie entre les paliers haut et bas en raison de la forme des fentes et de la position du capteur.

1.9. Mesure de la vitesse de rotation

- **Comptage des changements de valeur (les interruptions)**
  - Pour déterminer la vitesse, il faut compter le nombre de transitions (bas-haut) par unité de temps.
  - Il n'est pas envisageable de faire cela dans la loop car le programme sera surement occupé à autre chose pendant la plupart des transitions. Pour contourner de problème, nous allons utiliser les interruptions (enfin une seule pour un moteur).
  - Une interruption est un moyen de suspendre le programme principale pour exécuter une autre tache lors de l'apparition d'un évènement. Une fois cette tache accomplie, le microcontrôleur reprend l'exécution du programme principal là ou il s'était arrêté.
  - Le temps passé dans la tache liée à l'interruption doit être très court sinon, le microcontrôleur peut passer sont temps à exécuter l'interruption si elle apparait en permanence.

1.9. Mesure de la vitesse de rotation

- **Comptage des changements de valeur (les interruptions)**
  - La carte arduino possède plusieurs interruptions dont les 2 principales sont appelées INT0 et INT1 et sont liées respectivement aux IO n°2 et 3.
  - Pour déclarer l'utilisation d'un interruption ainsi que la fonction à utiliser, il faut utiliser : attachInterrupt(numéro, fonction, MODE);
  - « numéro » correspond au numéro de l'interruption et donc à l'IO (0 pour l'IO n°2 et 1 pour l'IO n°3), « fonction » est la fonction à appeler, MODE peut prendre 4 états en fonction de ce qu'il faut détecter : LOW, CHANGE, RISING (passe de LOW à HIGH), FALLING (passage de HIGH à LOW).
  - Pour cette exemple nous déclarons : attachInterrupt(0, changeD, RISING);
  - La fonction « changeD » ne reçoit aucun paramètre et n'en renvoie aucun. Par contre elle change la valeur d'un compteur (unsigned int) compteurD :

```
void changeD() {compteurD++;}
```

1.9. Mesure de la vitesse de rotation

- **Relevé du compteur à intervalles de temps réguliers et vitesse**
  - A présent, il faut relever la valeur du compteur à intervalles de temps réguliers pour déterminer la vitesse de la roue (m/s ou cm/s)
  - Pour cela, on peut utiliser la librairie TimerOne qui comme son nom l'indique utilise (modifie !) le timer n°1 de la carte arduino.
  - **Il faut alors avoir conscience que cela rend inutilisable les PWM des IO n°9 et 10 et que cela perturbera aussi la librairie « servo ».**
  - Pour l'utilisation de cette librairie et son téléchargement (qui peut aussi se faire directement avec le gestionnaire de librairie de l'IDE), il faut aller ici :
    - ✓ github.com/PaulStoffregen/TimerOne
    - ✓ playground.arduino.cc/Code/Timer1/

1.9. Mesure de la vitesse de rotation

- **Relevé du compteur à intervalles de temps réguliers et vitesse**
  - On déclare d'abord la librairie : `#include <TimerOne.h>`
  - Dans le setup, on initialise le temps de comptage (en µs), ici 150 ms, et on attache une fonction au timer (CalculVitesseD) qui permettra le calcul de la vitesse :

```
Timer1.initialize(150000);
Timer1.attachInterrupt(CalculVitesseD);
```

  - Reste à définir la fonction « CalculVitessD » :

```
void CalculVitesseD(void){
  vitesseD = Mult*compteurD;
  Serial.println(compteurD,2);
  compteurD = 0;}
```

  - « Mult » correspond à : (3.14 × 6)/(20 × 150000 × 0.000001 où 6 est le diamètre de la roue et 20 le nombre de fentes de la roue codeuse.

2. Servomoteur

2.1. Description

Qu'est ce qu'un servomoteur ?

- Un servomoteur ou plus simplement servo, est un moteur qui permet une rotation d'un certain angle compris entre 0° et 180°.
- Un servo se caractérise pas sa vitesse de rotation et son couple
- Il en existe une multitude en fonction de ces deux paramètres, de l'alimentation et de la qualité de la mécanique (plastique, métal, roulement…)

9 grammes
4.8 à 6 volts
Engrenages en plastique
0.12 sec/ 60°

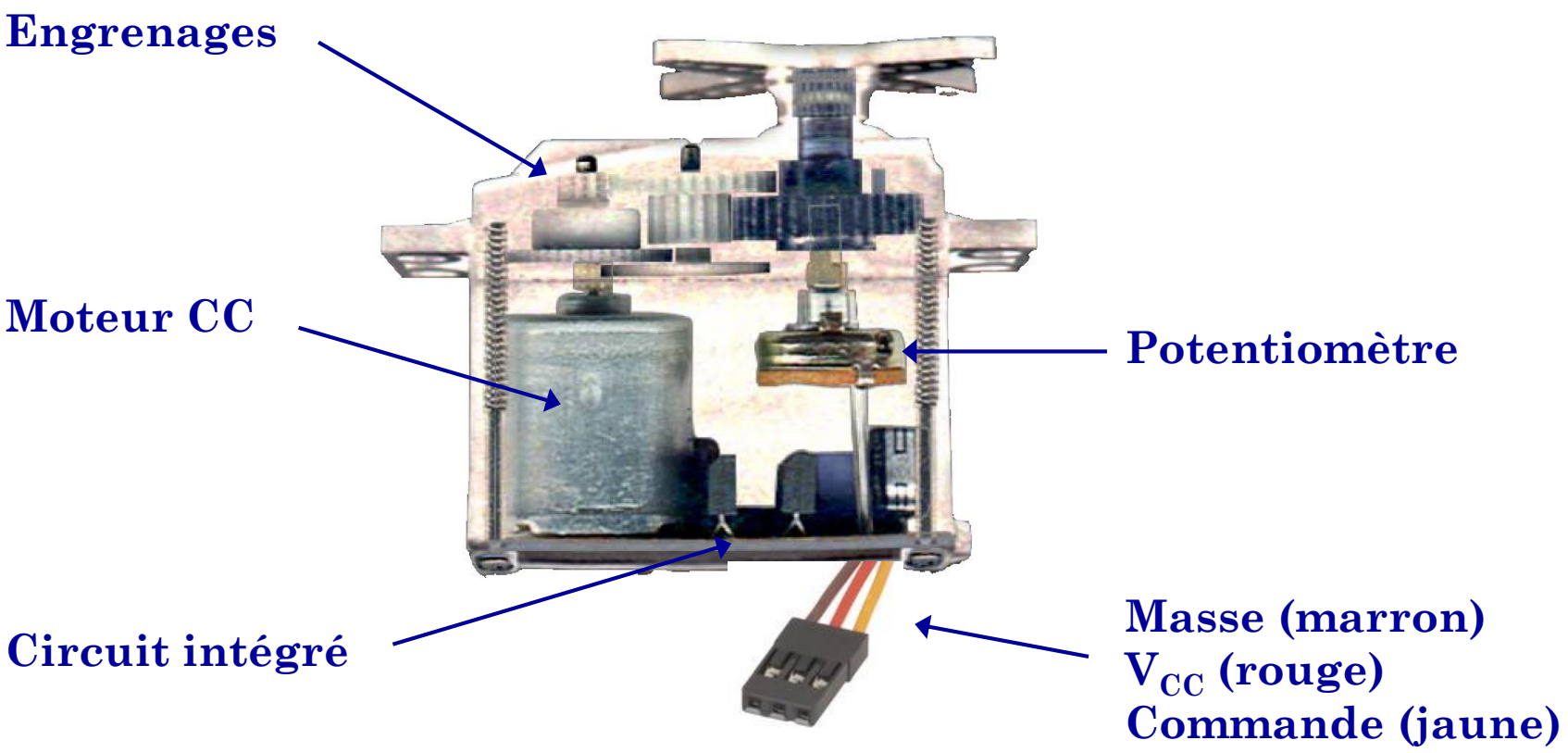
4.8 à 6 volts
roulement à bille
0.17 sec/60°
0.8 Nm

6 à 7.4 volts
0.14 sec/60°
1.67 Nm

2. Servomoteur

2.1. Description

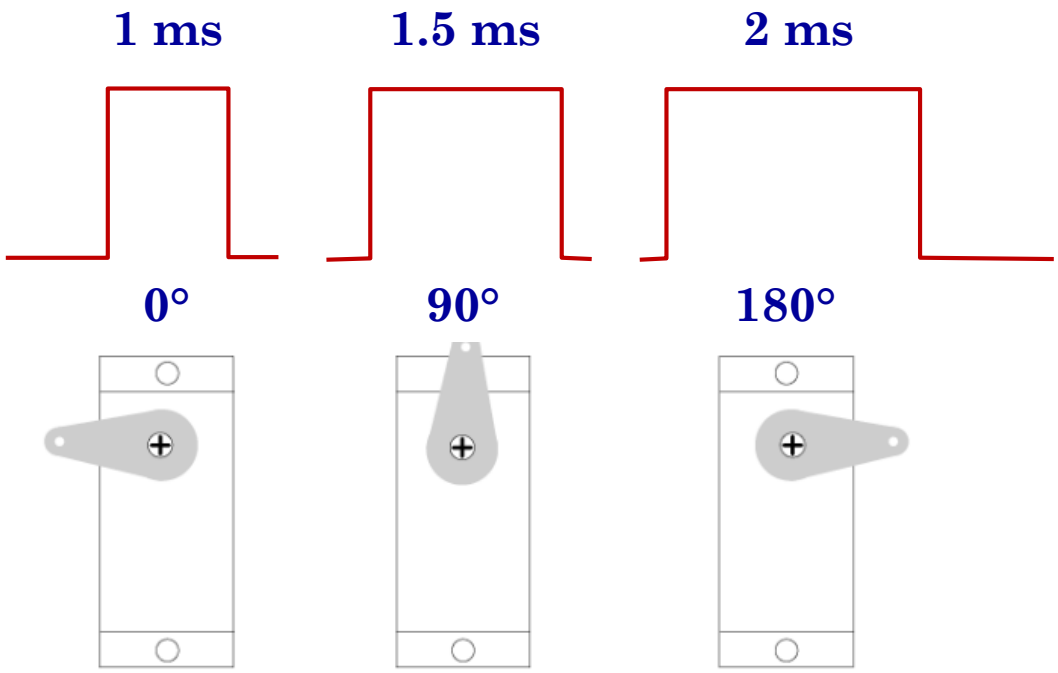
- **Eléments d'un servomoteur**



2.1. Description

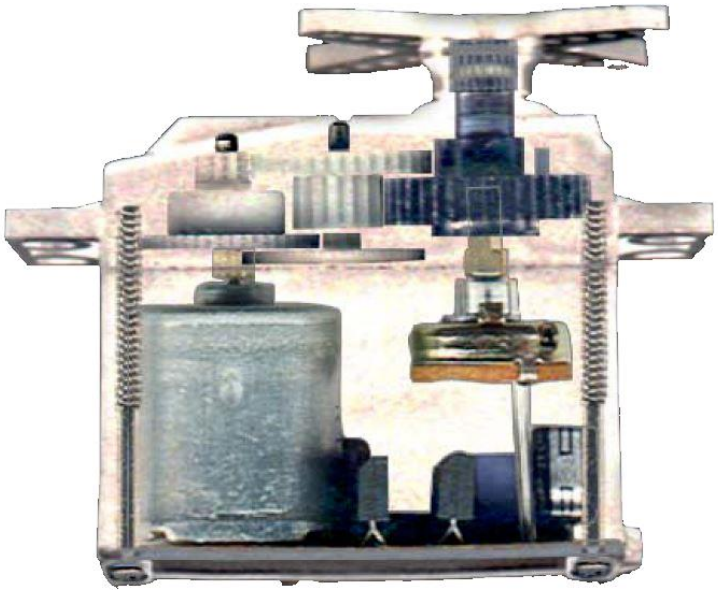
Fonctionnement

- Une impulsion de 1 ms correspond à un angle de 0 °
- Une impulsion de 2 ms correspond à un angle de 180 °
- Les impulsions doivent être séparée d'au moins 20 ms
- **Il faut en permanence envoyer l'impulsion pour atteindre l'angle puis le maintenir.**



2.1. Description

- **Fonctionnement**
  - Quand on envoie une impulsion au circuit intégré, il fait tourner le moteur dans la bonne direction
  - Les engrenages font aussi tourner le potentiomètre dont la résistance change
  - Lorsque que la résistance correspond à l'angle souhaité, le moteur s'arrête
  - Si l'axe change de position, le moteur se remet en marche pour maintenir l'angle



2. Servomoteur

2.1. Description

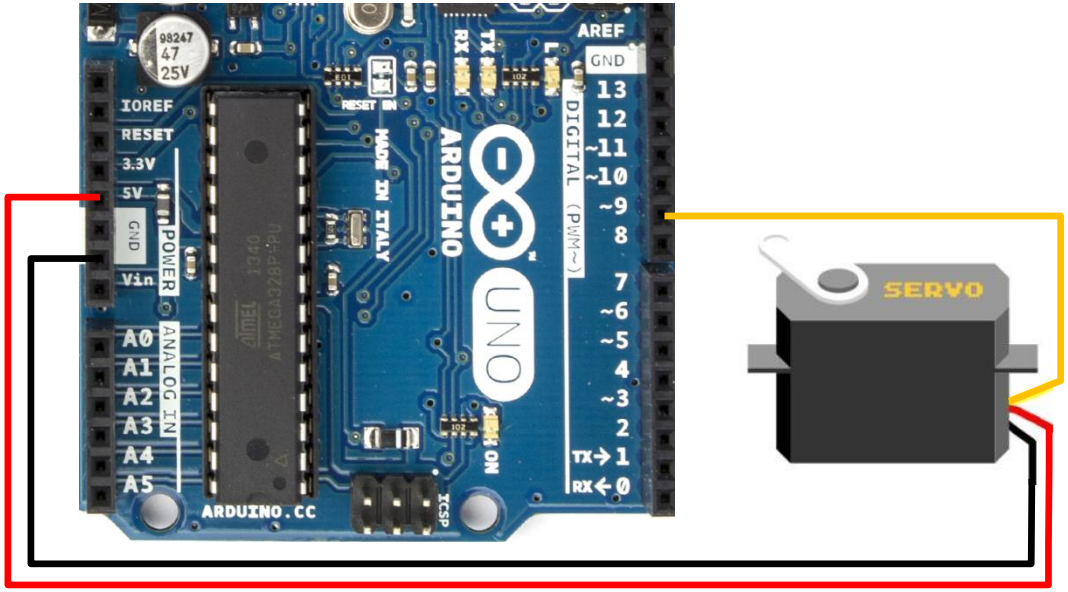
- **Composants supplémentaires**
  - Les éléments connectés à l'arbre du servo risquent de le tordre
  - On peut fixer un bloc porteur sur le servo qui empêchera cela

- Il est possible d'ajouter des engrenages pour augmenter le couple d'un servo

2.2. Mise en œuvre

Montage

- Le fil de commande d'un servomoteur est en général de couleur orange.
- En fonction du nombre de servomoteurs utilisés, il faut utiliser une alimentation extérieure et ne pas oublier de relier sa masse avec celle de la carte arduino.



2.2. Mise en œuvre

Librairie « Servo »

- Plutôt que de réaliser les signaux, il est parfois préférable d'utiliser une librairie qui s'occupera toute seule du problème
- On peut utiliser la librairie « Servo » qui est déjà intégrée au logiciel arduino et il faut le préciser en début de programme

```
#include <Servo.h>
```

- On déclare alors une variable qui correspond au servo que l'on va utiliser.

```
Servo servo1;
```

- Il faut alors préciser dans le « setup » sur quelle IO se trouve le servo. Dans cette exemple on choisit l'IO n°9 et il n'est pas nécessaire de préciser qu'elle sera utilisée comme une sortie

```
servo1.attach(9);
```

- Avec cette librairien, on peut contrôler jusqu'à 12 servomoteurs avec une carte UNO et 48 avec la carte MEGA.

2.2. Mise en œuvre

Librairie « Servo »

- La fonction .attach peut aussi prendre 3 arguments, où min est donné en microsecondes et correspond à l'angle minimum du servomoteur et max (en microsecondes) correspond à l'angle maximum

```
servo1.attach(9, min, max);
```

- Par défaut min = 544 et max 2400
- A noter qu'il est possible de détacher le servo de l'IO avec : servo1.detach();
- On peut alors choisir un angle pour le servo : Servo1.write(90);
- On peu aussi indiquer l'angle en microsecondes plutôt afin de gagner en précision puisque l'angle en ° doit être un entier :

```
Servo1.writeMicroseconds(Angle en microsecondes);
```

2.2. Mise en œuvre

- **Librairie « Servo »**
  - La librairie « Servo » est basée sur le Timer 1 qui est lui-même utilisé par la librairie « VirtualWire » qui sert en communication RF
  - Cela induit des conflits de librairies quand on veut télécommander un objet qui utilise des servos

2.2. Mise en œuvre

- **Librairie « Servo Timer 2 »**
  - La librairie « Server Timer 2 » a été développée pour contourner les problèmes de librairie entre les librairies « Servo » et « VirtualWire » et elle utilise le Timer n°2
  - Elle permet de contrôler jusqu'à 8 servos
  - Il faut télécharger et installer la librairie :

    github.com/nabontra/ServoTimer2
  - Il faut le préciser que l'on utilise cette librairie en début de programme

    `#include < ServoTimer2.h>`
  - On déclare alors une variable qui correspond au servo que l'on va utiliser :

    `ServoTimer2 servo1;`

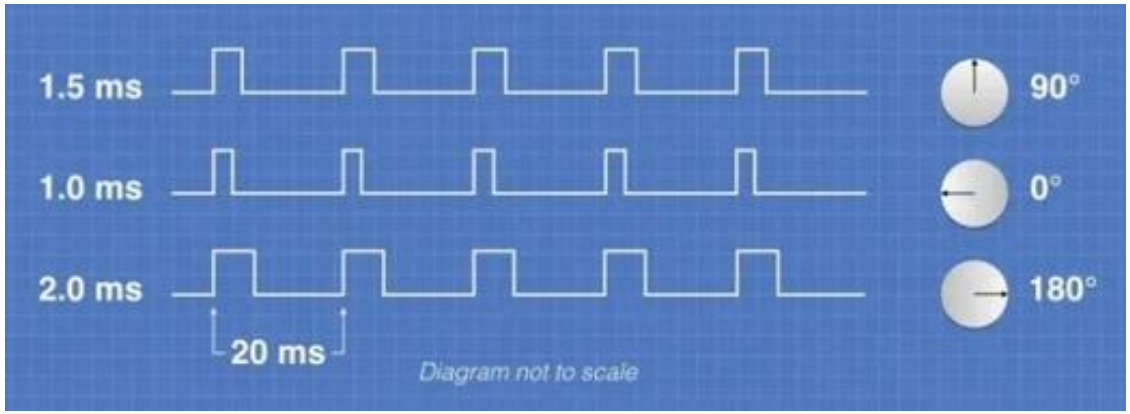
2.2. Mise en œuvre

- **Librairie « Servo Timer 2 »**
  - La suite est identique à la librairie « Servo » :

```
servo1.attach(9);

servo1.detach();

Servo1.write(90);
```

2.3. Les servomoteurs à rotation continue

- **Présentation**
  - Ces servomoteurs permettent de contrôler la vitesse de rotation du moteur.
  - On peut par exemple les utiliser à la place de moteurs CC pour faire un avancer une voiture.
- **Mise en œuvre**
  - Les fonctions à utiliser sont les mêmes que pour le servomoteur « normal ».
  - L'application d'un angle de 90° correspond cette fois à un moteur qui est à l'arrêt.



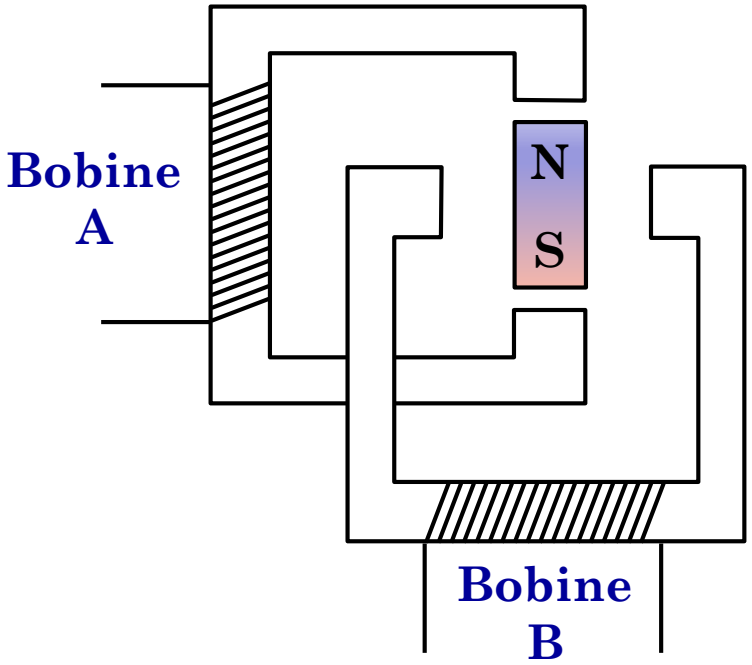
3.1. Description

- **Qu'est ce qu'un moteur pas à pas ?**
  - C'est un moteur qui est un compromis entre le moteur CC et le servomoteur
  - Il est en effet possible de le faire tourner à des vitesses variables et on contrôle précisément sa position si on connait sa position initiale
  - En effet et contrairement au servomoteur, il n'y a pas un capteur qui permet de connaitre la position angulaire du moteur
  - Il en existe plusieurs types en fonction du nombre de fils qui en sortent : 4 (moteur bipolaire), 5 et 6 (moteurs unipolaires ou à réluctance variable)
- **Dans quoi les utilise t'on ?**
  - On les trouve dans tous les objets qui nécessitent une rotation précise comme par exemple les imprimantes quelles soient pour le papier (ou pour le plastique (3D)

3.2. Type de moteur et fonctionnement

Moteur bipolaire

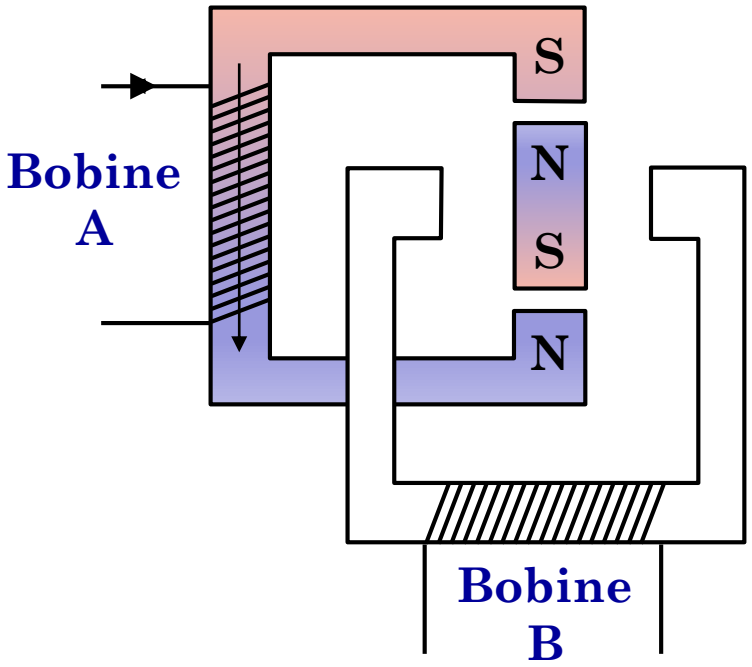
- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement



3.2. Type de moteur et fonctionnement

Moteur bipolaire

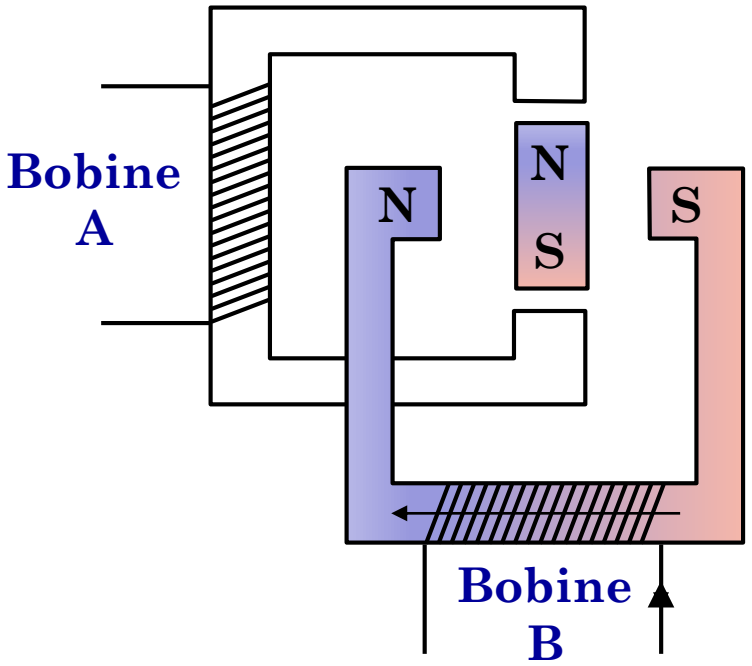
- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement
- En fonction du sens (bipolaire) du courant dans les bobines, elles peuvent créer un pôle nord ou sud



3.2. Type de moteur et fonctionnement

Moteur bipolaire

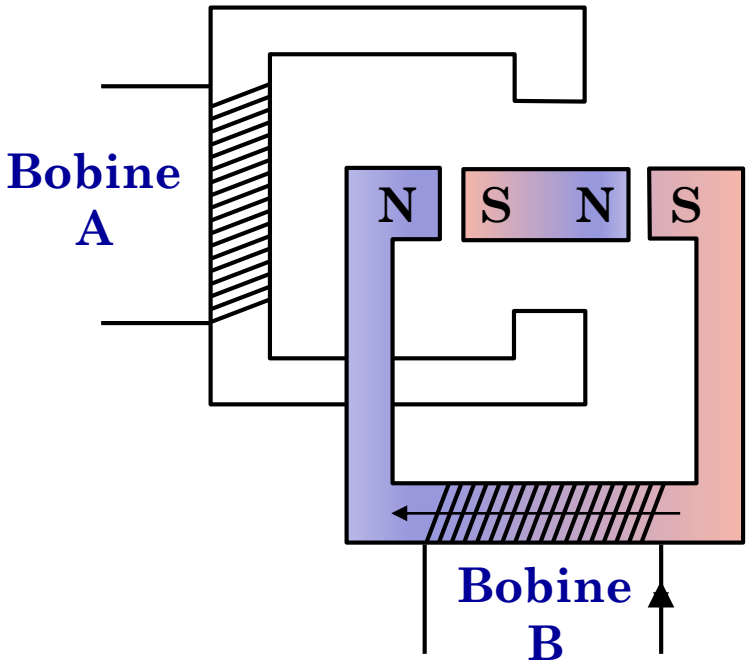
- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement
- En fonction du sens (bipolaire) du courant dans les bobines, elles peuvent créer un pôle nord ou sud



3.2. Type de moteur et fonctionnement

Moteur bipolaire

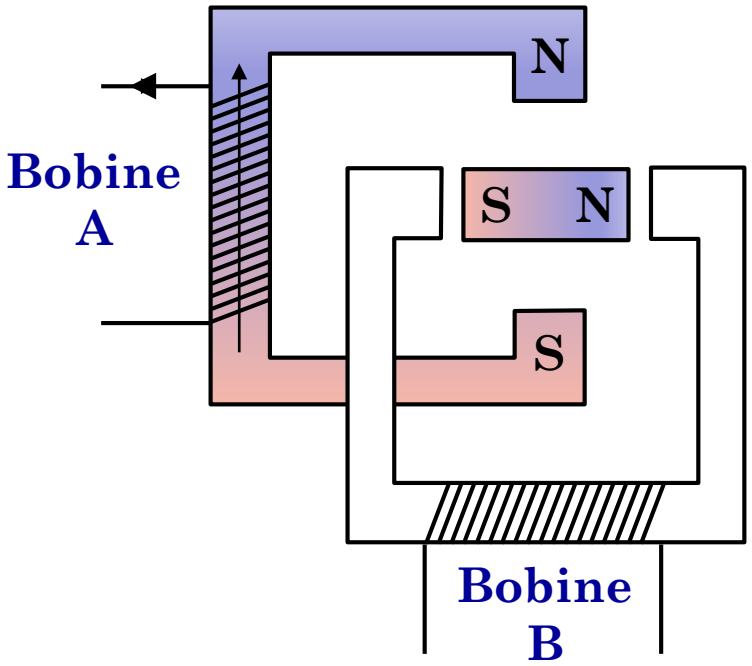
- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement
- En fonction du sens (bipolaire) du courant dans les bobines, elles peuvent créer un pôle nord ou sud
- Ce quart de tour correspond à un pas



3.2. Type de moteur et fonctionnement

Moteur bipolaire

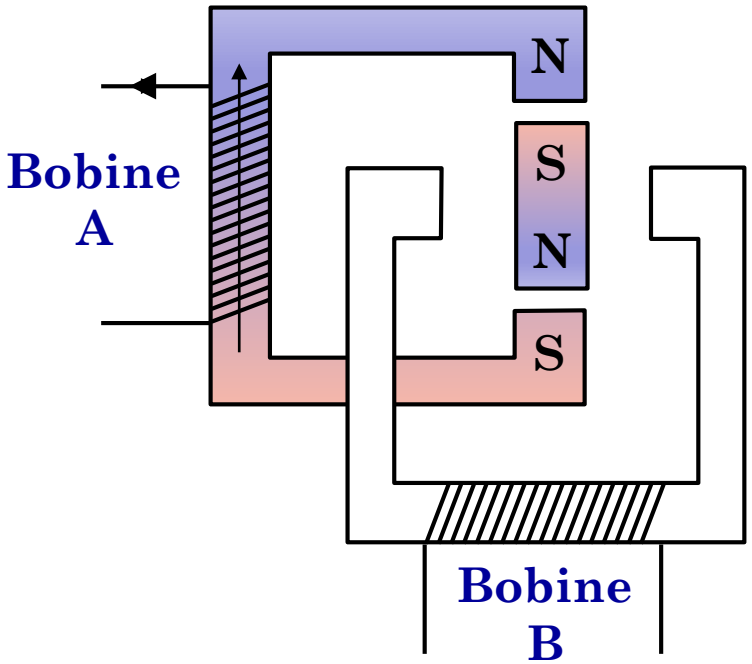
- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement
- En fonction du sens (bipolaire) du courant dans les bobines, elles peuvent créer un pôle nord ou sud
- Ce quart de tour correspond à un pas



3.2. Type de moteur et fonctionnement

Moteur bipolaire

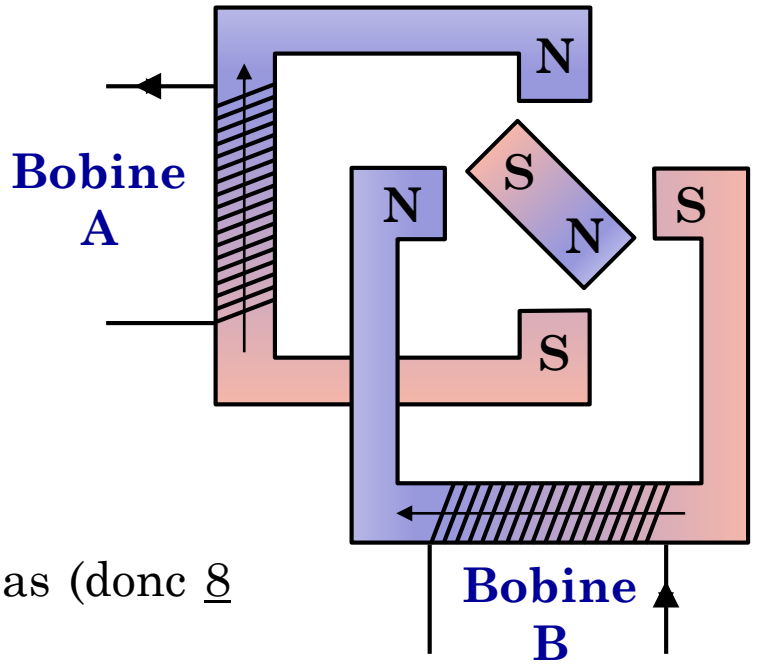
- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement
- En fonction du sens (bipolaire) du courant dans les bobines, elles peuvent créer un pôle nord ou sud
- Ce quart de tour correspond à un pas
- Dans cet exemple il faut 4 séquences (donc 4 pas) pour que le rotor fasse un tour complet.



3.2. Type de moteur et fonctionnement

Moteur bipolaire

- Ce moteur possède 4 fils pour alimenter les 2 bobines et un aimant
- En l'absence d'alimentation, le rotor peut tourner librement
- En fonction du sens (bipolaire) du courant dans les bobines, elles peuvent créer un pôle nord ou sud
- Ce quart de tour correspond à un pas
- Dans cet exemple il faut 4 séquences (donc 4 pas) pour que le rotor fasse un tour complet.

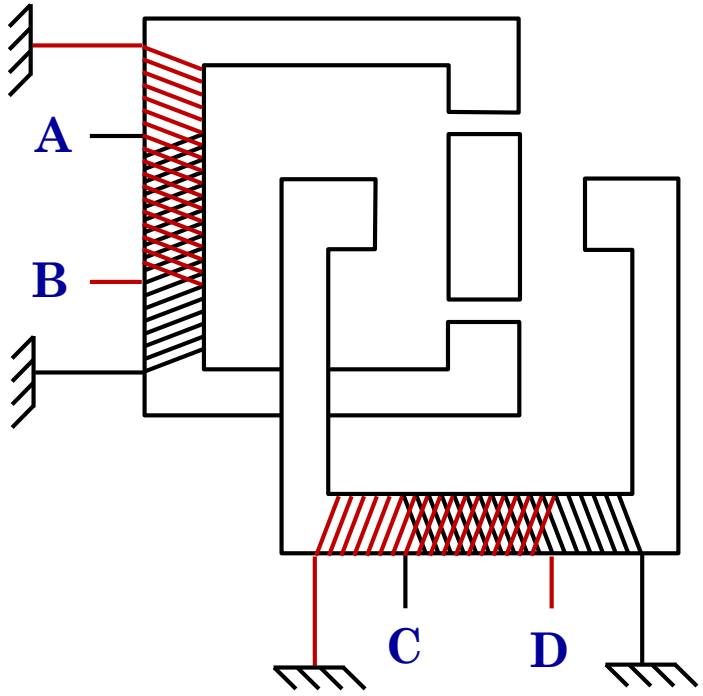


- On peut multiplier par 2 le nombre de pas (donc 8 séquences) en alimentant les 2 bobines
- Il existe des moteurs qui ont beaucoup plus de pas : 24, 48 …

3.2. Type de moteur et fonctionnement

Moteur unipolaire

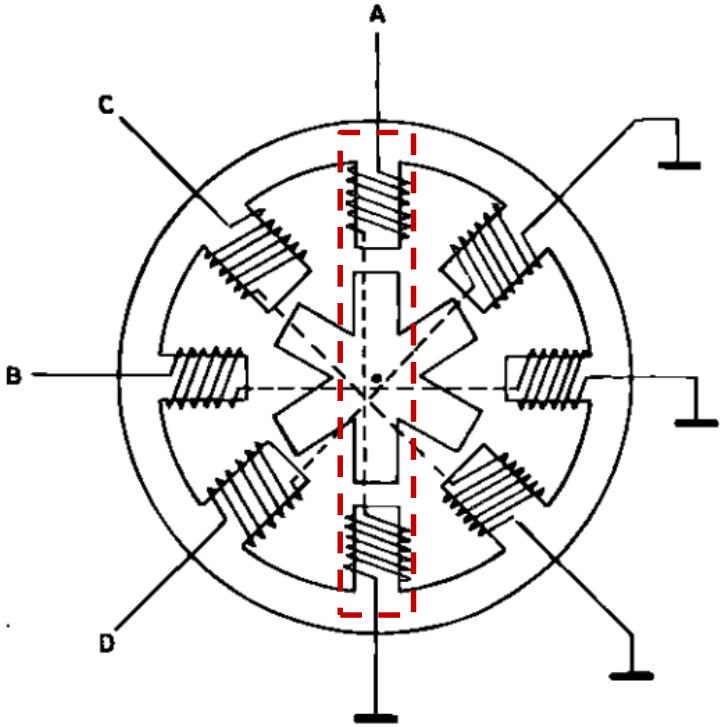
- Ce moteur possède 4 fils pour alimenter les 4 bobines et 1 aimant
- On alimente les 4 bobines successivement et le courant ne passe que dans un seul sens (unipolaire)



3.2. Type de moteur et fonctionnement

Moteur à réluctance variable

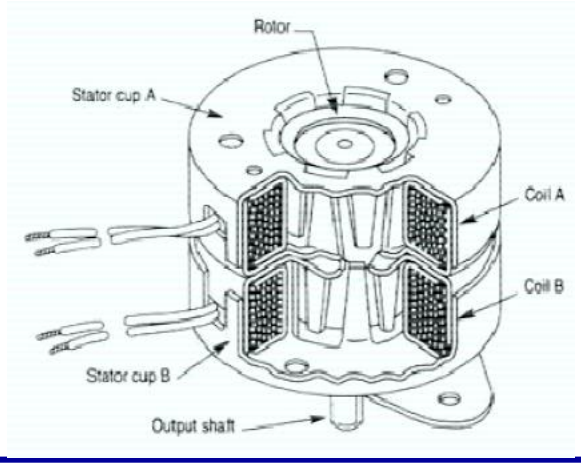
- Il est constitué de 4 bobines et de fer doux pour le rotor en remplacement de l'aimant
- Le fer doux conduit très bien le champ magnétique et cherche toujours à s'aligner avec lui

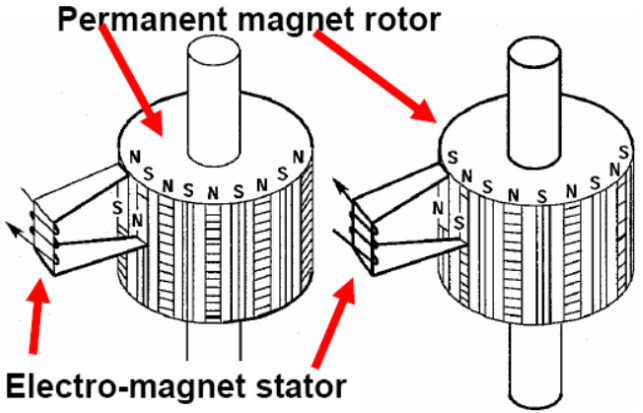


3.3. Exemple du moteur 28ybj-48

Présentation du moteur

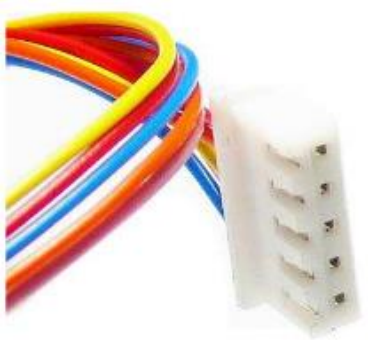
- C'est un moteur unipolaire avec réducteur (1:64) qui s'alimente sous 5 V
- Il est alimenté par 5 fils (4 bobines + fil commun)
- Le couple est très faible : 34.3 mNm
- En mode 4-séquences, il faut 32 séquences pour faire un tour et avec le réducteur on trouve qu'il faut 32 × 64 = 2048 pas pour faire un tour ce qui donne un angle par pas de 360 / 2048 = 0.1758 °

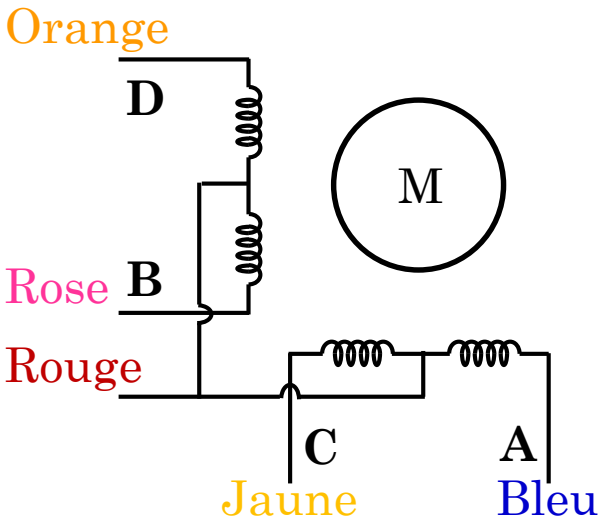




3.3. Exemple du moteur 28ybj-48

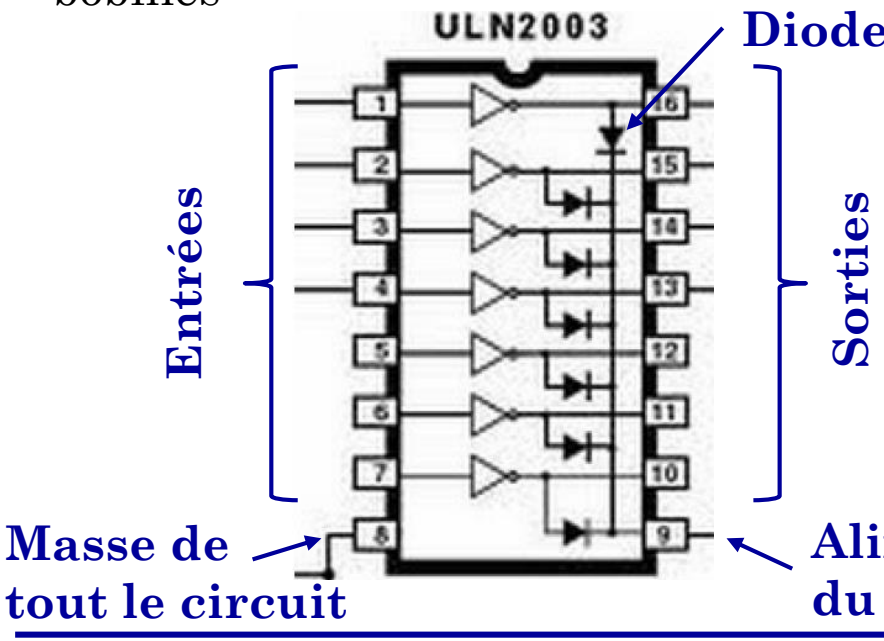
- **Mise en mouvement du moteur**
  - Les sorties des bobines sont repérées par les couleurs des fils
  - En mode 4-séquences et dans le sens horaire, il faut suivre la séquence : AB-BC-CD-DA
  - En mode 4-séquences et dans le sens horaire, on peut aussi suivre la séquence : A-B-C-D mais le couple est 2 fois moins fort
  - En mode 8-séquences et dans le sens horaire, il faut suivre la séquence : A-AB-B-BC-C-CD-D-DA

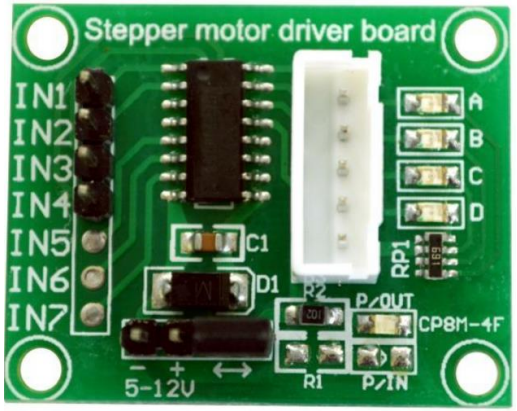




3.3. Exemple du moteur 28ybj-48

- **Montage**
  - Le moteur est fournit avec une carte comportant le circuit ULN2003 qui se compose de 7 transistors Darlington avec diodes de roue libre
  - Des LED indiquent quelles sorties (i.e. Darlington) délivrent du courant
  - Les entrées IN permettent d'activer les sorties et donc d'alimenter les bobines

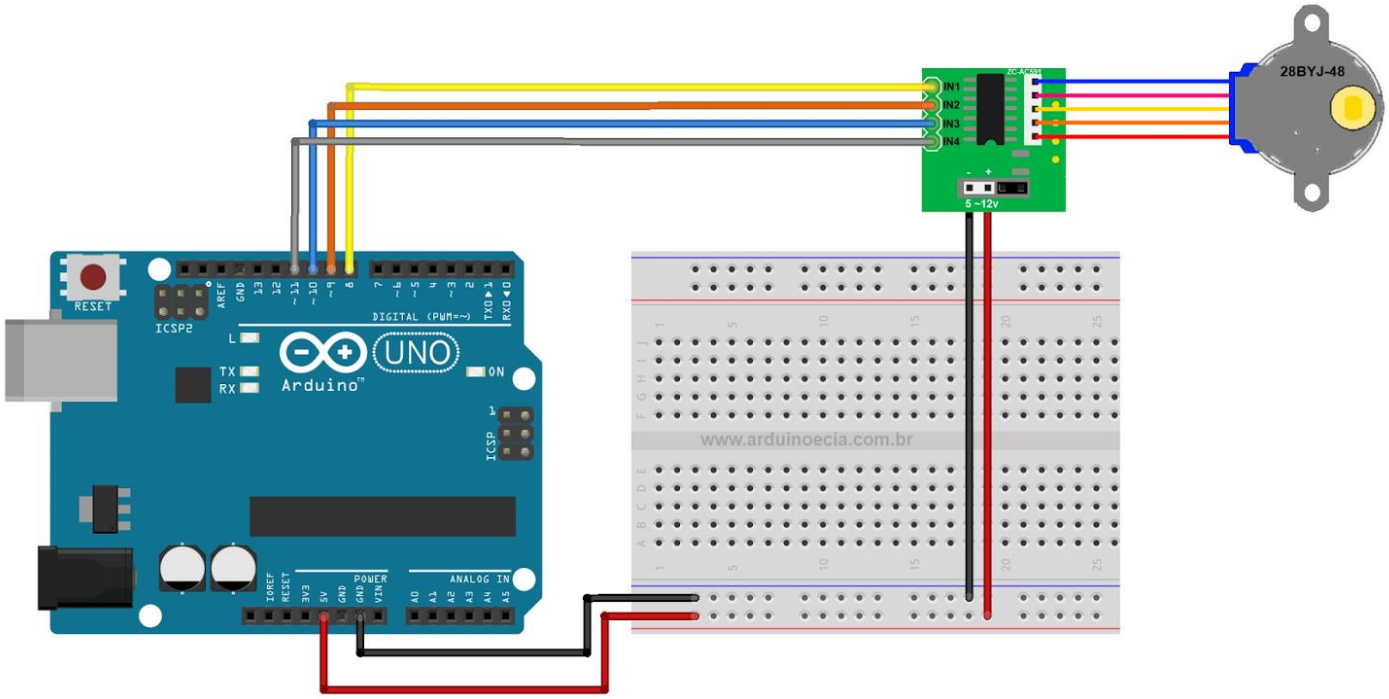




3.3. Exemple du moteur 28ybj-48

Montage

- Il est toutefois préférable de choisir une alimentation extérieure pour le moteur et il ne faut pas oublier de connecter sa masse avec celle d'Arduino



3.3. Exemple du moteur 28ybj-48

- **Programme**
  - Exemple de 4 séquences A-B-C-D avec un tour complet
  - Pour 1 tour : 32 (séquences) / 4 (A-B-C-D) × 64 (réducteur) = 512 boucles

```
#define IN1  2
#define IN2  3
#define IN3  4
#define IN4  5
int temps =10;
int i=0;
void setup(){
  Serial.begin(115200);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  for (i=0; i <= 511; i++){
    Serial.println(i);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    delay(temps);

    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    delay(temps);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    delay(temps);

    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    delay(temps); }}
void loop() {}
```

3.3. Exemple du moteur 28ybj-48

- **Programme**
  - Le temps entre chaque séquence doit être d'au moins 1740 µs donc le moteur peux faire un tour en : 1.74 (ms) × 2048 (pas) = 3.563 s
  - Donc il peux faire au mieux 16.8 tours par minute (= 60 / 3.563)

```
#define IN1  2
#define IN2  3
#define IN3  4
#define IN4  5
int temps =10;
int i=0;
void setup(){
  Serial.begin(115200);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  for (i=0; i <= 511; i++){
    Serial.println(i);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    delay(temps);

    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    delay(temps);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    delay(temps);

    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    delay(temps); }}
void loop() {}
```

3.3. Exemple du moteur 28ybj-48

- **Librairie « stepper »**
  - Cette librairie (déjà intégrée au logiciel arduino) permet de contrôler les moteurs unipolaire et bipolaire
  - Il déclarer l'utilisation de la librairie en début de programme

```
#include <stepper.h>
```

  - On définit alors les connections du moteur et le nombre de pas (nombrepas) pour un tour en créant un objet « monmoteur »

```
Stepper monmoteur(nombrepas, PIN1, PIN2);          Bipolaire
Stepper monmoteur(nombrepas, PIN1, PIN2, PIN3, PIN4);   Unipolaire
```

  - Pour le moteur unipolaire, la librairie impose l'alimentation de 2 bobines à chaque pas (donc : AB-BC-CD-DA)

3.3. Exemple du moteur 28ybj-48

- **Librairie « stepper »**
  - Il existe aussi une fonction pour la vitesse (vitesse) de rotation (en rpm)

    monmoteur.setSpeed(vitesse)
  - On indique alors le nombre de pas (pas) que l'on souhaite :

    monmoteur.step(pas)
  - Et pour faire tourner le moteur dans l'autre sens :

    monmoteur.step(–pas)

3.3. Exemple du moteur 28ybj-48

- **Librairie « stepper »**

```
#include <Stepper.h>

const int nombrePas = 32*64;
Stepper monMoteur(nombrePas, 2, 3, 4, 5);

void setup() {
  monMoteur.setSpeed(1);}

void loop() {
  monMoteur.step(nombrePas);
  delay(500);

  monMoteur.step(-nombrePas);
  delay(500);}
```

2048 pas

ordre : IN1, IN2, IN3, IN4

1 tour par minute, on ne peut pas faire moins sauf en diminuant « nombrePas »

1 tour complet dans le sens horaire

1 tour complet dans un sens anti-horaire

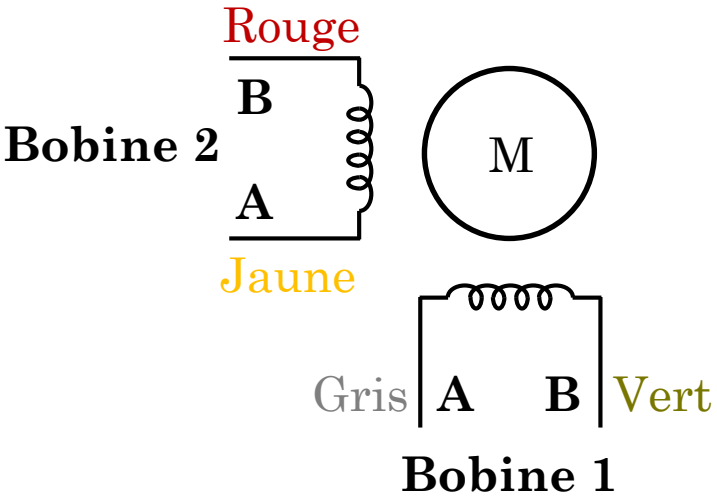
- On peut aussi s'orienter vers une autre librairie qui offre plus de possibilités

www.airspayce.com/mikem/arduino/AccelStepper/

3.4. Exemple du moteur NEMA17

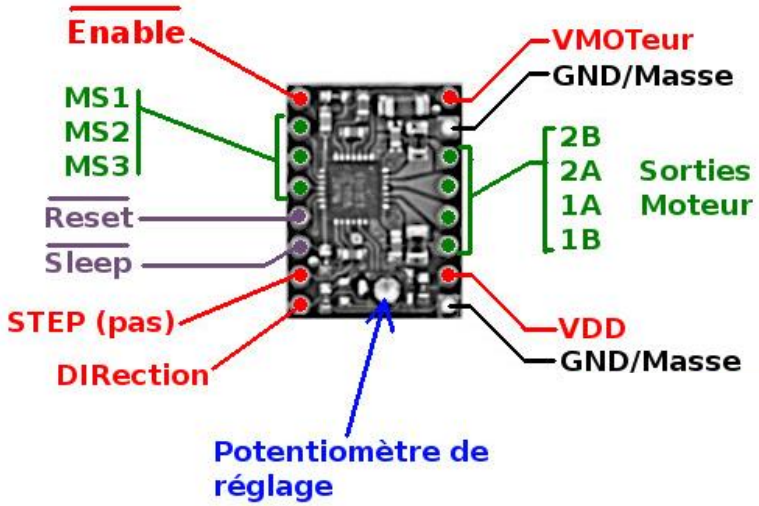
Présentation du moteur

- NEMA est l'abréviation de « National Electrical Manufacturers Association»
- Il mesure 1.7" de côté (soit 4.3 cm). Il existe aussi les NEMA 23
- C'est un moteur bipolaire qui s'alimente en 12 V pour obtenir un couple de 20 N.cm. On peut utiliser une tension plus faible mais le couple chute
- Le pas est de 1.8 °



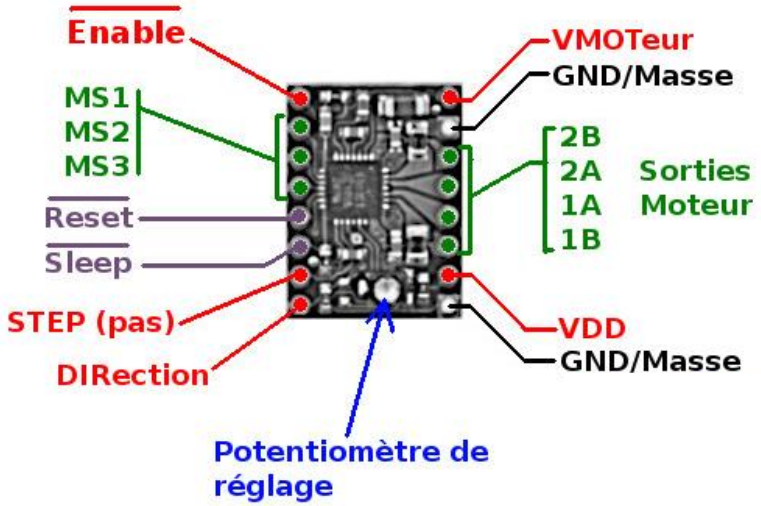
3.4. Exemple du moteur NEMA17

- **Mise en œuvre avec le driver A4988 : description**
  - Ce driver (ou contrôleur de moteur pas-à-pas) permet un contrôle très simple du moteur
  - Avec un radiateur, il peut faire passer jusqu'à 2 A
  - Il faut lui connecter l'alimentation de l'Arduino (VDD et GND) et celle du moteur (VMOTeur et GND)
  - L'entrée Enable est active à l'état bas et par défaut (sans rien connecter) elle est à l'état bas
  - L'entrée Reset est active à l'état bas et par défaut (sans rien connecter) elle est à l'état haut



3.4. Exemple du moteur NEMA17

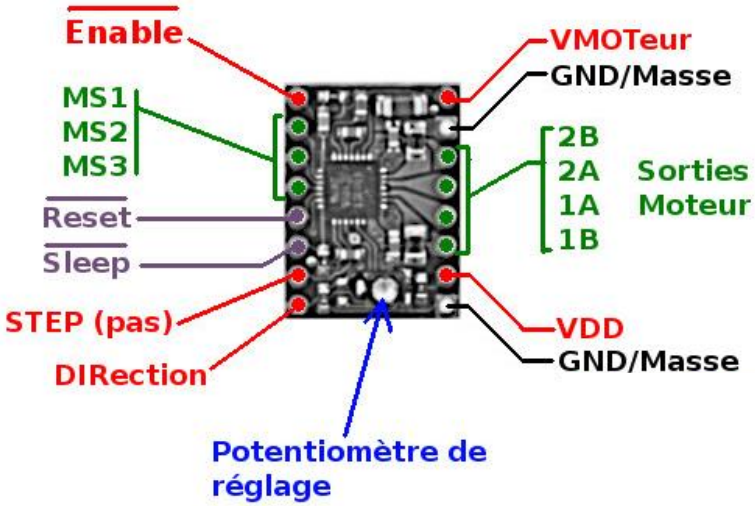
- **Mise en œuvre avec le driver A4988 : description**
  - L'entrée Sleep est active à l'état bas et permet de faire rentrer le module en mode basse consommation. On peut la connecter à l'entrée Reset pour qu'elle soit toujours inactive
  - L'entrée STEP permet de faire tourner le moteur d'un pas a chaque passage à l'état haut. Cela signifie qu'un simple générateur de signal carré permet de faire tourner le moteur
  - L'entrée DIR permet de sélectionner le sens de rotation (état haut pour le sens horaire)



3.4. Exemple du moteur NEMA17

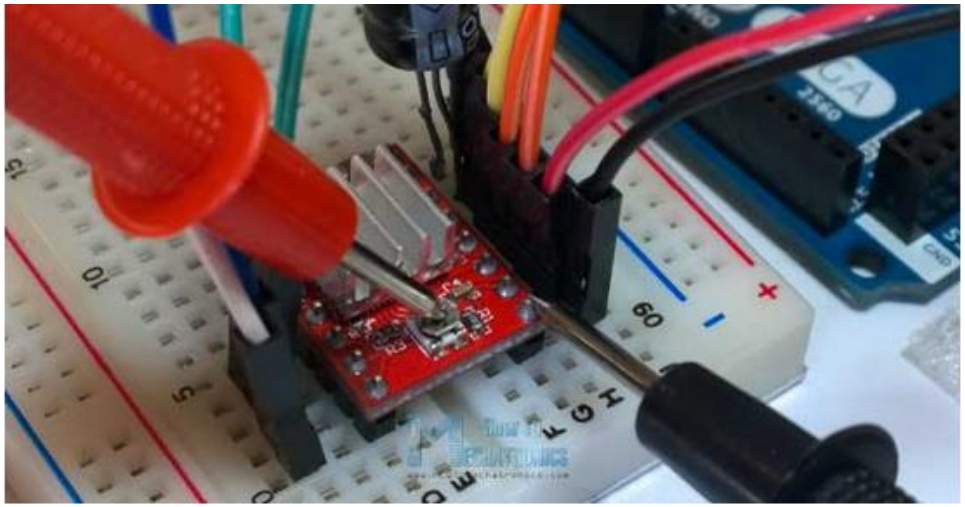
- **Mise en œuvre avec le driver A4988 : description**
  - Les entrées MS1 à 3 permettent de sélectionner la résolution du pas en accord avec la table de vérité
  - Par défaut les entrées MS1 à 3 sont à l'état haut ce qui sélectionne le mode « Full »

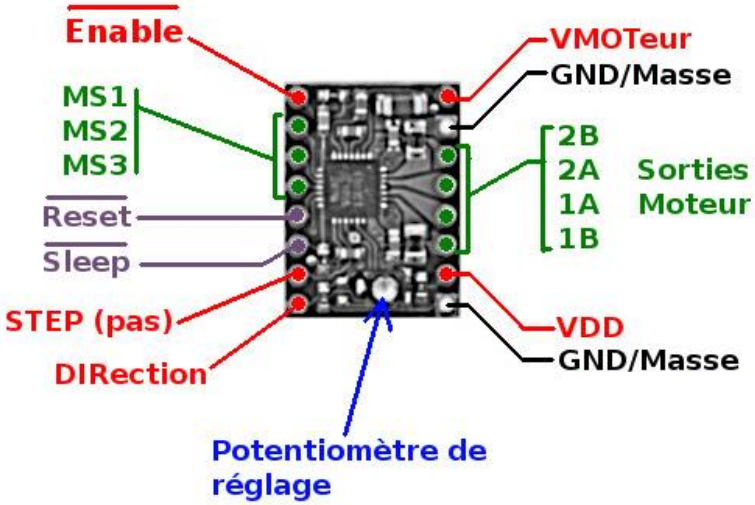
MS1	MS2	MS3	STEPS
L	L	L	FULL
H	L	L	HALF
L	H	L	QUARTER
H	H	L	EIGTH
H	H	H	SIXTEENTH



3.4. Exemple du moteur NEMA17

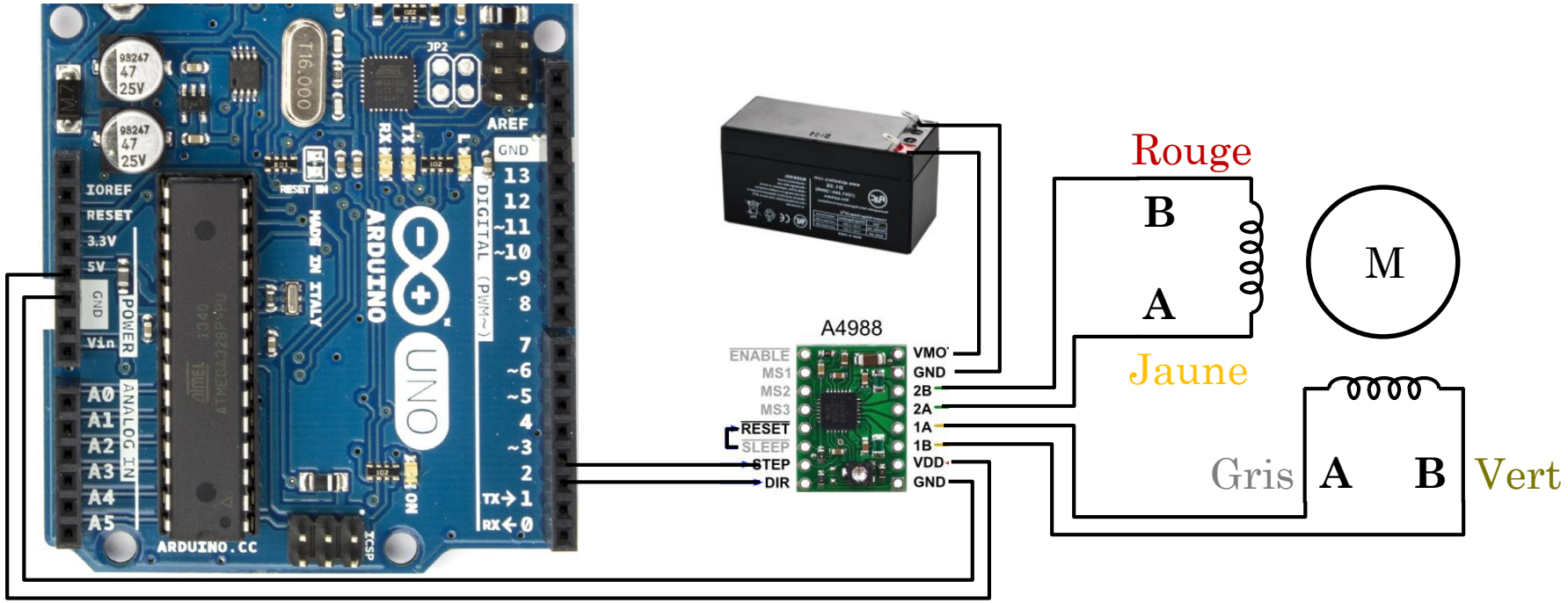
- **Mise en œuvre avec le driver A4988 : description**
  - Un potentiomètre permet d'ajuster le courant maximum qui passera dans les bobines et ainsi ne pas dépasser le courant limite du moteur
  - Dans le cas du NEMA 17, ce courant limite est de 350 mA
  - La tension entre la borne « + » du potentiomètre et la masse est donnée par : courant = 2.5 × tension. Pour courant = 0.35 A il faut obtenir tension = 0.14 V





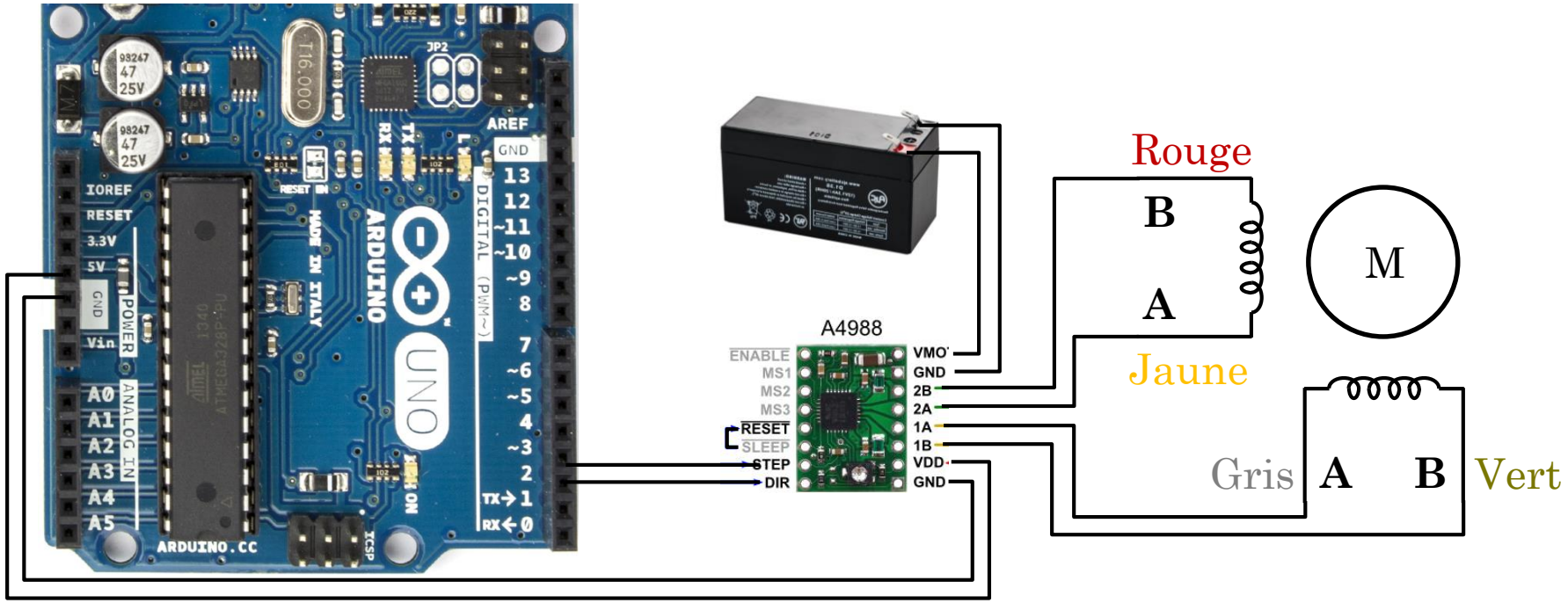
3.4. Exemple du moteur NEMA17

- **Mise en œuvre avec le driver A4988 : montage**
  - Le montage nécessite une capacité de découplage pour l'alimentation du moteur afin de lisser les appels de courant des bobines



3.4. Exemple du moteur NEMA17

- **Mise en œuvre avec le driver A4988 : montage**
  - A noter : la couleur des fils dépend du moteur et du constructeur ! On peut utiliser un ohmmètre pour identifier (court circuit ou circuit ouvert) les fils des 2 bobines. Si le moteur ne tourne pas (alors que le temps entre chaque pas est « long »), c'est qu'il faut permuter les fils d'une des bobines.



3.4. Exemple du moteur NEMA17

- **Mise en œuvre avec le driver A4988 : programme**
  - Avec ce programme, le moteur fait un tour complet et attend 1 s avant de recommencer
  - Le délais (ici 4 ms) entre chaque pas est à régler, il faut laisser le temps au moteur de réaliser le mouvement !

```
const int Pas = 3;
const int Dir = 2;
void setup() {
  Serial.begin(115200);
  pinMode(Pas,OUTPUT);
  pinMode(Dir,OUTPUT);
  digitalWrite(Dir,HIGH);
}
```

```
void loop() {
  for(int x = 0; x < 200; x++) {
  digitalWrite(Pas,HIGH);
  delayMicroseconds(500);
  digitalWrite(Pas,LOW);
  delay(4);
  }
delay(1000);
}
```

4.1. Description

- XXX

EN COURS D'ECRITURE